DESIGN AND IMPLEMENTATION OF AN SDLC LIKE LOCAL AREA LOOP NETWORK

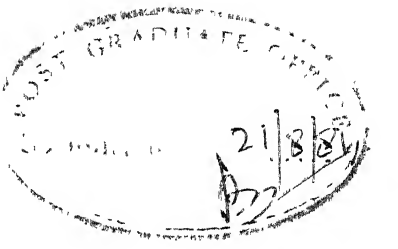
A thesis submitted
in Partial Fulfilment of the Requirements
for the degree of

MASTER OF TECHNOLOGY

by
S. BANDYOPADHYAY

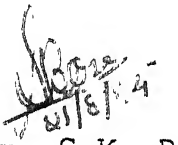
to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
AUGUST 1984

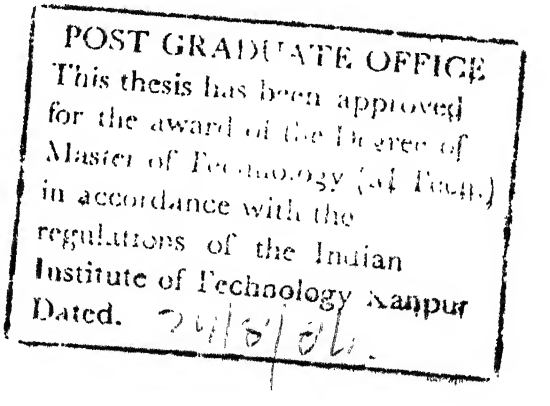


CERTIFICATE

It is to certify that this work entitled, 'DESIGN AND IMPLEMENTATION OF AN SDLC LIKE LOCAL AREA NETWORK' by S. Bandyopadhyay has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree.



Dr. S.K. Bose
Assistant Professor
Department of Electrical Engineering
I.I.T. Kanpur.



ACKNOWLEDGEMENTS

I would like to place on record my sincerest thanks and appreciation to my thesis supervisor Dr. Sanjay K. Bose, whose influence and encouragement have contributed greatly to the success of this project. Any attempt on my part to express my acknowledgement to him would only be fatuous.

I take this opportunity to express my heartfelt thanks and regard to Dr. R.N. Biswas for making available the various facilities of the department without which this project could not have been implemented.

I am grateful to Messrs. P. Malhotra and B.V. Reddy, Research Engineers of the Microprocessor Development Systems Lab., for their constant cooperation and helpful advice during the period of my work there.

To my friends, Mukund, Deshpande, Ravi Kumar, Biswajit and Gayaprasad, I express my thanks for the many fruitful discussions that I had with them and for making the memory of the last two years pleasant by their constant company.

Special thanks goes to K.K. Islam and M. Chakrabarty for rendering their helpful hands at various stages of the documentation.

Finally, but not the least, to Mr. C.M. Abraham for his rapid and neat typing of the manuscript of this thesis.

S. Bandyopadhyay
Aug.18,1984

I.I.T. Kanpur.

ABSTRACT

In this thesis, the development of a microprocessor based intelligent interface for handling the data link layer protocol in a Multiaccess Multidrop Network is described. The development effort has culminated into the production of an interface board which has the data source/destination on one side and the data network on the other. Apart from specifying the data destination, the user of this interface board will not be required to handle any of the operational aspects of the network itself.

The system currently implemented is a modified SDLC loop network capable of supporting one primary station and upto 254 secondary stations. The network allows data link connections between any two stations in the network. For testing purposes the data rate supported by the loop is presently fixed at 9.6 Kbps. Very minimal changes in hardware will allow upgrading the loop speed to 64 Kbps. This interface is being implemented using standard LSI chips and other off-the-shelf components. As a next planned upgrade the speed can be improved to 880 Kbps by changing some of the LSI chips (and some of the interfacing software). A possible extension is to develop the necessary hardware and software for a gateway interface for communicating with another identical loop network. Another likely upgradation is to enable the data link interface to support multiple users. Outline schemes for these upgradations are also described.

CONTENTS

CHAPTER 1

INTRODUCTION

1.0 THE CONCEPT OF NETWORKING

Computer Networking was born of the marriage of the field of communications with that of computers. This concept has transformed the way computer systems are organized these days. It has opened up vistas of new possibilities of the use of computer systems in different walks of life. The old model of a centralized computer serving all the needs of an organization is rapidly being replaced by the idea of having a large number of separate but interconnected computers to do the same job more efficiently. These systems are called Computer Networks.

The basic theme behind all kinds of computer networks is that of distributing the intelligence of a centralized system. Based on this idea we have a wide variety of Computer Networks, varying in the degree of cohesiveness of the network as well as the mode of interaction among the network elements.

There are several factors which makes the idea of networking advantageous over that of centralized systems: Firstly, the need to make all programs, data and other resources available to the user without regard to the physical location

of the resource and the user. Secondly, to provide a high degree of reliability by having alternative means of supporting a system in case of hardware failure. With a network, the temparary loss of a computing element gets reflected over the system as a corresponding reduction in the efficiency of the system. On the other hand, in a centralized system the whole unit gets crippled because of this. There is an important third factor which relates to the relative price of computing versus communication. Until about 1970, computers were relatively expensive compared to communication facilities. The reverse is now true. As a result instead of transmitting data generated at widely scattered points to a central computer, it has become more lucrative to process the data where it is captured and only send occasional summaries to the computer centre. This reduces the communication cost, which now represents the larger percentage of the total cost involved in the whole operation. Lastly, the advent of small computers with much superior price/performance ratio over the large ones, also favours networking. Main frames are roughly a factor of 10 faster than the largest single chip microprocessors but they cost a 1000 times more. This imbalance has urged designer's to harness a collection of microcomputers to outperform the large mainframes at lower cost.

1.1 LOCAL AREA NETWORKS

Local area networks are those which interconnect a collection of processors in close geographical proximity.

There are mainly two reasons behind the establishment of a local network. Firstly, to allow a collection of computers, terminals and peripherals located in the same building or adjacent buildings not only to intercommunicate but also to access a remote host or another network. In the absence of the local network, separate connections would be needed between the remote facility and each of the local machines. The local network enables the remote facility to tap onto the local network in one place. Secondly, the local networks exploit the advantages of functionally distributed computing. Here, some of the machines are dedicated to perform specific functions, such as file storage, data base management, terminal handling and so on. By having different machines perform different tasks, the goal is to make the implementation simpler and more efficient.

There are three main types of local networks : 1) Carrier sense, 2) Loop and, 3) Shared Memory. Carrier sense and Loop networks are conceptually similar though technically different. Carrier sense type of networks may use radio or cables - either twisted pair or coaxial - just like loop networks, but the organization is fundamentally different. Whereas a carrier

sense network (using cables) is basically a passive, electrically connected cable onto which all stations tap, a loop net is actually a series of point-to-point cables between consecutive station. Also the interfaces used on loop networks are active rather than passive. The shared memory systems are both conceptually as well as technically different from the other two types, since all processes in the entire system can share a common address space if they so desire.

1.2 THE SDLC PROTOCOL

The SDLC is a bit oriented Data-Link level protocol for local area network, designed by IBM. SDLC transmits data as a serial bit stream of any bit length, from 0 data bits to the largest number of bits that a memory can handle. There are no implied character boundaries within the bit stream, which is called an information field.

Since bit oriented protocols do not have to contain an integral number of characters, a new method is needed to delimit frames yet preserve data transparency. The data transparency method used by SDLC is called Bit-Stuffing. Each frame begins and ends with a special bit pattern, namely 01111110, called Flag. Whenever the transmitting hardware encounters more than five consecutive ones in the data stream, it automatically stuffs a Ø bit into the outgoing bit stream. When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it

automatically destuffs (i.e. deletes) the 0 bit. Bit stuffing is completely transparent to the software in both the sending and receiving computers.

The information field is always preceeded by an address field and a control field. In SDLC protocol the address field and the control field are each exactly eight bits wide. The address field is primarily of importance on multidrop lines, where it is used to identify one of the stations. It may also be used in a point-to-point line to distinguish between command and response frames. The control field is used for specifying the type of frame: information, acknowledgement or command.

The information field is followed by a 'Frame Check' field. The frame check character in SDLC is 16 bits long. It is derived mathematically from the bit pattern of the combined address, control and information field. It is actually a minor variation of the well-known cyclic redundancy code, using CRC-CCITT as the generating polynomial. The variation is to allow lost flag bytes to be detected.

Figure 1.20 describes the frame structure used in SDLC protocol.

The SDLC protocol when implemented in a communication network assumes the presence of one primary station and one or more secondary stations. Secondary stations communicate

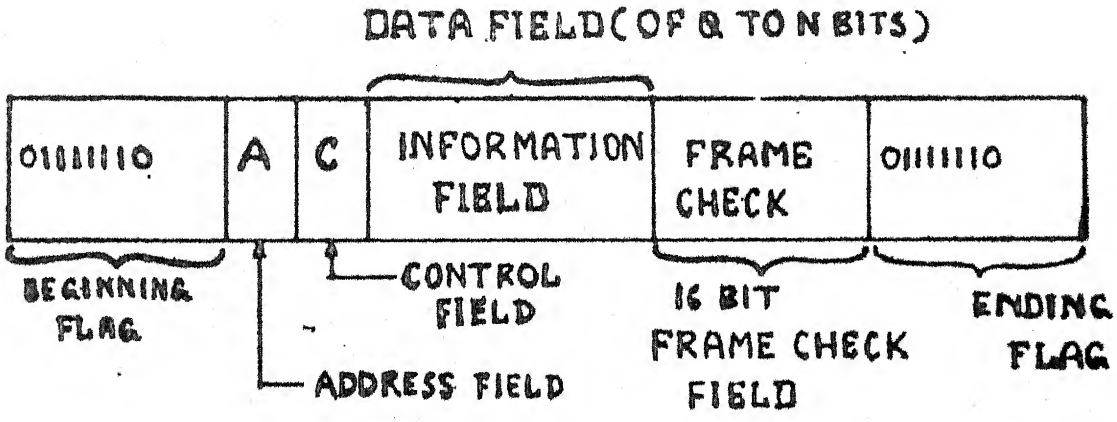


FIG: 1.20 FRAME IN SDLC PROTOCOL

with the primary station. Secondary stations do not communicate directly among themselves but via the primary only. Fig. 1.21 shows the different network configurations supported by SDLC.

The organization of SDLC and its frame structure has some nonobvious repercussions listed below :

1) The Address field of a frame will always be the address of a secondary station. The Primary station uses the address field of a transmitted frame to identify an intended secondary recipient, since there can be more than one secondary recipient The secondary station uses the address field of a transmitted frame to identify itself as the transmitter. Since the secondary can transmit directly only to the primary, it does not require to identify the destination, but for the sake of the destination, it needs to identify the source.

2) All informations must be transferred between stations as frames. Therefore, some frames will transfer control information rather than data, while other frames will transfer data. Data frames will also have accompanying control information contained in the control field.

3) A receiving station uses the frame check field in order to accept or reject the entire frame. SDLC protocol operate on the basis of error detection on received frame and retransmission on a frame rejected due to error.

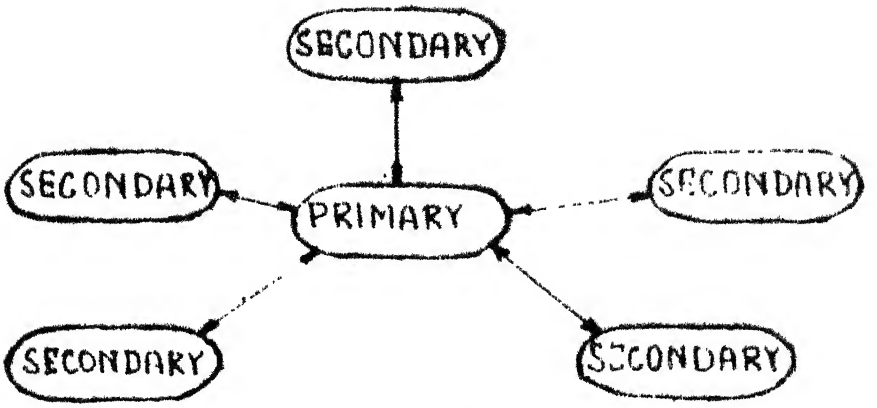


FIG: 1.21a POINT-TO-POINT NETWORK

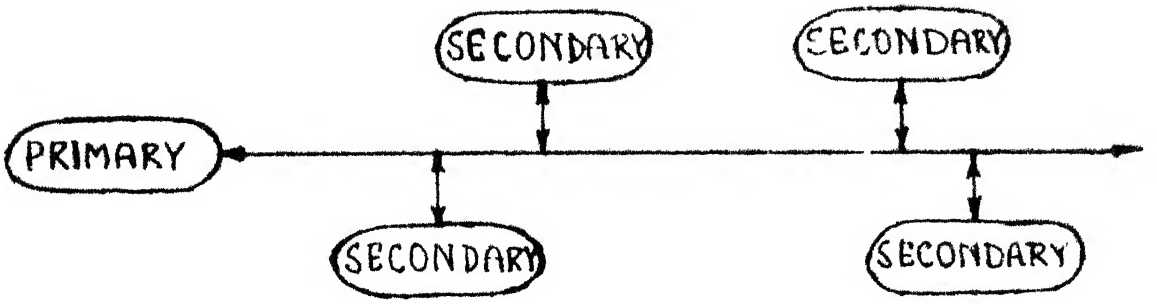


FIG: 1.21b MULTIPOINT NETWORK

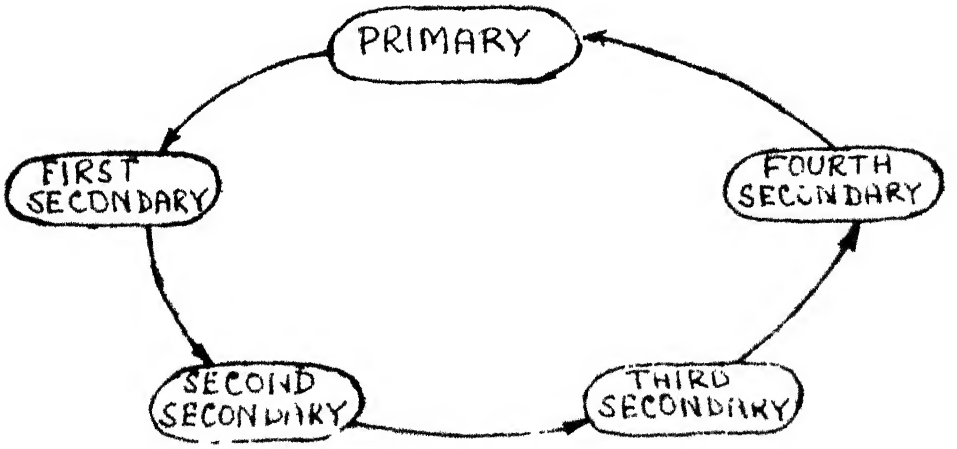


FIG: 1.21c LOOP NETWORK

4) No data breaks are allowed within a frame. In between frames a transmitter may either output a sequence of flag characters or it may output a continuous high signal. The continuous high signal is called an idle signal.

5) SDLC protocol imposes no constraints on the data contained in a frame's information field. So far as the protocol is concerned, the information field is an unstructured serial sequence of bits. It is upto the transmitter and receiver to mutually agree on how the bits must be interpreted (if at all). It makes no difference to the protocol whether data is being transferred as bytes, words or amorphous bit streams, it also makes no difference whether the data is ASCII characters or a memory object code dump.

6) SDLC protocol allows a serial data stream to be encoded and decoded using NRZ (positive) or inverted (NRZI) logic. Positive logic interprets a high signal as a 1 bit and a low signal as a 0 bit. NRZI interprets a constant signal level as a 1 bit and a changed level as a 0 bit. Figure 1.22 illustrates the two different logic interpretation for the same serial signal.

7) The bit stuffing technique in addition to serving the data transparency purpose is also helpful in clock recovery in case where the hardware supporting SDLC recovers the clock from received stream of data. Due to zero bit stuffing a transition

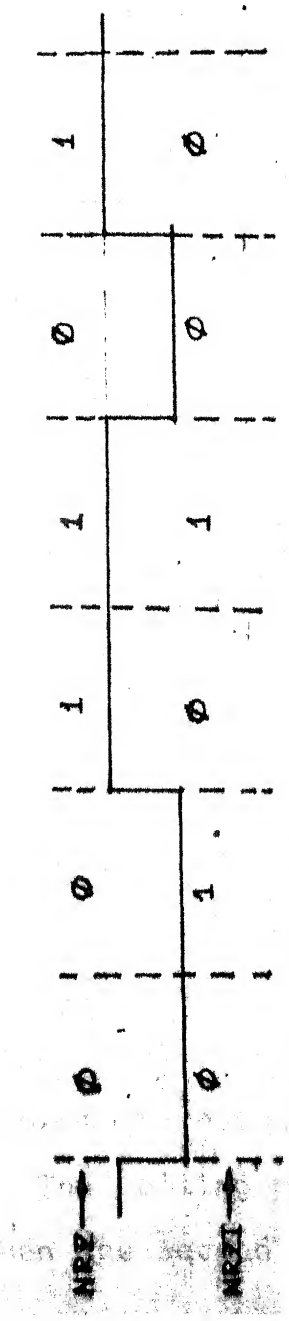


FIG: 1.22 NRZ AND NRZI LOGIC INTERPRETATION OF THE SAME SERIAL SIGNAL

in the data is assured after at least every 5 bits. The DPLL used in the receiver hardware is able to synchronize itself with the help of these data transitions.

1.3 THE SDLC LOOP

In SDLC mode, secondary stations can be configured to form a loop with the primary, rather than as point to point or multipoint, as shown in Fig. 1.21. In the SDLC loop the primary transmits frames to the first secondary in the loop. The secondary retransmits the frame downstream. Of necessity, the secondary introduces a one bit delay when retransmitting a received frame. Each secondary in the loop receives the frame from the immediately upstream secondary; after the additional delay it retransmits the frame. Thus the primary will ultimately receive back any frame it transmits. The secondary receives only those frames whose address fields match with its own address.

The SDLC loop is made contention free by the scheme of token passing. In a loop a station cannot start transmitting a frame if it is receiving a frame from an upstream station or is likely to be interrupted by an upstream station's transmission. In case of that occurring there would be a collision. To avoid this problem the primary transmits a polling pulse around the loop. The polling pulse is received by the first secondary, and then the second secondary, and so on until it

returns to the primary. The polling pulse is called an 'END OF PROCESS' (EOP) character, it is formed by the trailing of a frame's closing flag followed by seven 1 bits. This is illustrated in Fig. 1.30a.

When the first secondary receives an EOP character, it has the option of transmitting the EOP character downstream to the second secondary, or it can transmit a frame before retransmitting the EOP character to the second secondary. After passing the EOP character on to the second secondary, the first secondary cannot transmit a frame again until the primary sends another EOP character around the loop.

When the secondary receives the EOP character, it can either transmit the EOP character directly to the third secondary, or it can transmit a single frame before retransmitting the EOP character to the third secondary. In this fashion the EOP character makes its way around the SDLC loop. The primary will ultimately receive back its polling frame, any frames that secondaries had to transmit, and finally the EOP character. Fig. 1.30b illustrates a possible sequence when secondaries 2 and 3 have something to transmit but not secondaries 1 and 4.

Thus in the SDLC loop protocol the probability of collision is totally absent. Moreover, configuring the stations as a loop has the major advantage that though physically data flows through the loop in only one direction, logically the loop

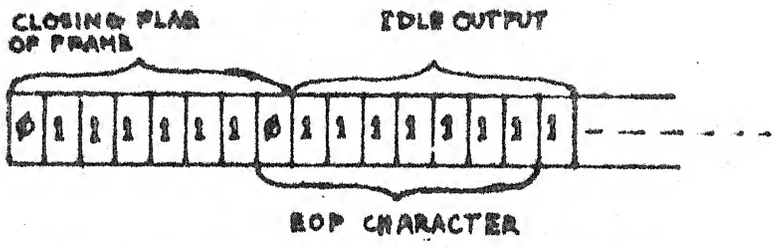


FIG: 1.36.a GENERATION OF EOP CHARACTER

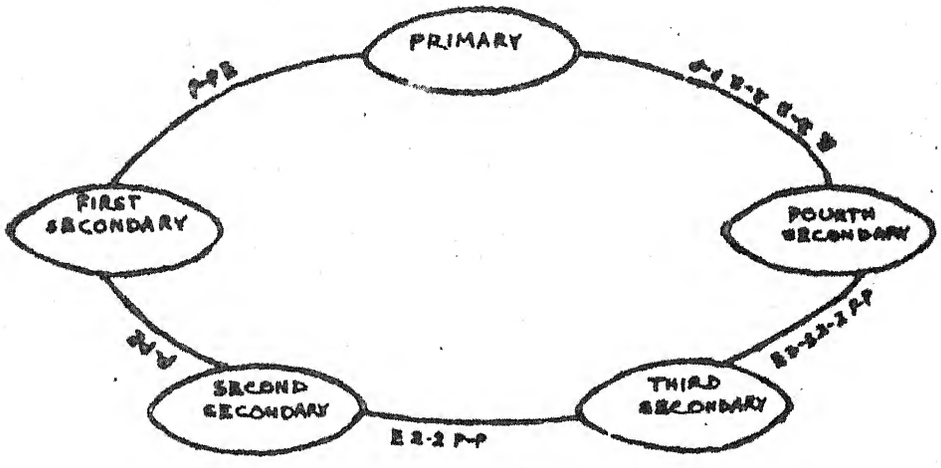


FIG: 1.36.b POLLING SEQUENCE IN THE LOOP

IN THE ABOVE FIGURE:

P-P REPRESENTS THE POLLING FRAME.

2-2 REPRESENTS THE FRAME TRANSMITTED BY THE SECOND SECONDARY.

3-3 REPRESENTS THE FRAME TRANSMITTED BY THE THIRD SECONDARY.

E IS THE EOP CHARACTER.

performs duplex operation. Since control of data flow in one direction is much simpler than that of bidirectional data, this configuration achieves simplicity without loss of efficiency.

1.4 THE IMPLEMENTATION OF SDLC IN THE CURRENT SYSTEM

The material covered in the last two sections describes the bare format of the SDLC protocol. However, its actual implementation has to take care of other things like 1) control frame manipulation (2) Acknowledgement scheme and (3) Secondary to secondary communication on top of the above structure. The basic structure of the SDLC does not get affected by the way these, strategies are evolved. But in their implementation of the protocol IBM has given concrete schemes for all of these and they are meant to be followed as part of the standard SDLC protocol implementation. However, for our current system we have only followed the bare format of SDLC; for the remaining we have implemented our own schemes keeping in view the particular environment in which the system is being developed. We now describe in some detail these features.

(1) Control frame manipulation : As we have already mentioned, an SDLC frame has an 8-bit control field. There is no restriction, as such, on what is to be sent on the control field, as the receive hardware does not examine this field. However, it is evident that as in SDLC all communications between the sending and the receiving stations have to be in the form of frames, we need a frame

qualifier field. The control field essentially carries that information. In other words, corresponding to any particular frame we would have to generate an 8-bit control code to specify what frame it is. We can thus have at most 256 different types of frames. In addition to the above obvious function, the control field can also be used to carry some hand-shaking information between the sender and the receiver.

In our implementation, we have divided the frames to be transmitted into four major categories. They are as follows :

1) Information frame
2) Acknowledgement frame
3) Polling frame
4) Supervisory/Command frame.

Table 1.40 illustrates the complete list of the frames designed.

The three main types of information frames in the loop are : Secondary to Secondary Information, Secondary to Primary Information and Primary to Secondary Information. Besides these, the Primary to Primary Information frame has been designed mainly to test the working of the loop. The other information frame designed is Data frame through gateway. Note that Bit 7 of the control code is used as an information for the receiver about the sender. If Bit 7 = 0, the sender is secondary, whereas if bit 7 = 1, the sender is primary.

Name of Frame	General Type	Control Code
Primary to secondary Information	Information	80H
Secondary to Primary Information	Information	01H
Primary to Primary Information	Information	81H
Secondary to secondary Information (from Secondary)	Information	02H
Secondary to Secondary Information (from Primary)	Information	82H
Primary to Secondary Acknowledgement	Acknowledgement	83H
Secondary to Primary Acknowledgement	Acknowledgement	04H
Primary to Primary Acknowledgement	Acknowledgement	84H
Secondary to Secondary Acknowledgment (from Secondary)	Acknowledgement	05H
Secondary to Secondary Acknowledgement (from Primary)	Acknowledgement	85H
Polling frame	Supervisory	86H
Reset a Secondary	Supervisory	87H
Disconnect a Secondary	Supervisory	8AH
Online Secondary	Supervisory	8BH
Status Poll of a Primary/Secondary	Supervisory	08H/88H
Extended Address Data frame through gateway	Information	0C/8C
Extended Address Acknowledgement frame through gateway	Acknowledgement	0D/8D

Tabl3 1.40

List of Frames Designed

Corresponding to the information frames we have the various Acknowledgement frames viz. secondary to secondary ACK, secondary to Primary ACK., Primary to Secondary ACK. and ACK. frame through gateway. There is a point to note about the ACK. frames. They may or may not have data associated with them. Depending on the code of the ACK. frame the data associated with it would be interpreted differently. We discuss this in Secondary to Secondary frame processing.

The Polling frame, though itself a supervisory frame is listed separately because of its special contribution to the operation of the loop. Of the remaining possible codes a few only are designed. More and more control codes can be generated for a complex environment involving diverse kinds of network elements which are to be controlled correctly for proper operation of the network.

(2) Acknowledgement Scheme : For acknowledgement we have used the simplest stop and wait protocol. This means a sender transmits a frame, waits for the acknowledgement to come back. It can send the next frame only after receiving the ACK. for the last frame. We have used positive acknowledgement scheme, so that 'no acknowledgement' means the information frame has been lost. The sender times out on acknowledgements and on a timeout it retransmits the frame until the retransmission threshold is exceeded. We have adopted a very simple but logical scheme for calculating the timeout interval,which makes it independent

of variable parameters like number of nodes connected to the loop or size of the message. Since we have an EOP passing network, and the EOP character circulates around the loop we use this to calculate the timeout period for the primary. According to the protocol described above the frames sent by the secondaries are all inserted between the polling frame and the EOP character. So at the time of receiving an EOP character the primary checks if it has got the ack. of the frame last transmitted, on getting no ACK, it updates an acknowledgement counter. A threshold (3 in our case) is set to that counter. If that threshold is exceeded ack. timeout process is invoked. This ensures that timeout does not get affected by the loop parameters. For secondaries we adopt a similar scheme. Since the secondaries do not generate interuupts on EOP, we count the number of polling frames received by a secondary since the last frame was transmitted, rather than the number of EOPs. A similar threshold is set on the polling frame counter which when exceeded enables the ack. timeout process to be invoked.

(3) Secondary to Secondary Communication : This is another essential feature which the SDLC protocol does not directly provide for. We describe it briefly here. A secondary to secondary communication according to our schemes involve the transmission of a secondary to secondary frame by any secondary and the reception of the frame by the intended recipient

secondary, the transmission of an ack. frame by the recipient secondary and the reception of the ack. frame by the transmitting secondary. However, secondaries cannot directly send frames to each other. So naturally both the original and the ack. frame has to be routed through the primary. From these we can infer the following requirements for an SS frame :

1) It must supply the destination address of the frame implicitly to the primary.

2) The primary must know whether the frame has come first time or after rerouting.

We satisfy both the criteria by the following way. When the original S-S-I frame come to the primary it has a code 02. The primary on checking this code decides that a SSI frame has come, for the first time. This is evident because any frame transmitted through Primary would have Bit 7 of control field = 1. The primary then accesses the first byte of data. The secondary adds the implied destination address as the first data byte. So the primary obtains the destination address simply by accessing the first data byte. It then does the following :

a) It switches the actual address and the implied address, so that the actual address (sender address) now comes to the 1st byte of data and the implied address comes to the address field of the frame.

b) It sets Bit 7 of the control code so that control code now becomes 82.

The destination secondary now receives a frame with control code = 82. It decides that a secondary to secondary frame (routed through primary) has come. It accepts the frame and sends an ack. frame with code = 05. The primary gets back first the frame with control code = 82, decides that it is a rerouted SSI frame and rejects it. It then gets the SSA frame with code = 05. It decides that a SSA frame has been received, for the first time. Exactly as before it does operations a) and b) on the received frame and retransmits it. The retransmitted frame has a control code = 85, and a destination address same as that of the sender secondary. The sender secondary receives the SSA frame with code = 85 and decides that the acknowledgement to its last frame has come. The Primary now gets back the routed frame of code = 85 and rejects it.

Thus secondary to secondary transmission is performed without affecting the basic SDLC structure, simply by checking and manipulation of the frame control byte. It has however the drawback of being slow because of the indirect transfer and the time taken for rerouting in the primary. Also the probability that an SS frame gets lostor corrupted is double that of ordinary direct frames. However, the scheme is quite effective and simple.

CHAPTER 2

HARDWARE DESIGN OF THE CONTROLLER

2.0 SCHEMATIC OF HARDWARE DESIGNED

The first criterion for the hardware design of a node is to achieve near total compatibility of the primary with the secondary. Essentially this ensures that in case of a failure in the primary, any of the secondaries can be transformed into a primary with a minimum change of hardware. This also saves effort in the fabrication stage in terms of p.c.b. making.

The design of the whole circuit can be functionally broken down into that of four parts. a) The Link Controller Unit, b) Message Storage and Processing Unit, c) Host Interface Unit, d) Timing and Interrupt Generation Unit.

As the name implies the part a) interacts with the Link and does the following :

1) Checks in serial data from the link at the correct baud rate, synchronously with the link and transforms it into parallel bytes of data.
2) Looks for address match in the received bit stream in its receive logic.
3) Does C.R.C. checking of the received bit stream in its receive logic.

4) Generates handshaking signals to the message processing unit (viz. part b)

5) Sends out serial data at programmed baud rate, after transforming from parallel bytes.

6) Generates a C.R.C. pattern for the transmitted string of data.

7) Attaches data headers (viz. Address, Control bytes programmed by user)

8) Attaches frame delimiters (viz. flag characters)

9) Sends out token (viz. EOP character) for the loop under program control.

The last activity is a job peculiar to the loop primary. In our hardware most of these operations are taken care of by the SDLC protocol controller chip 8273 (Intel).

Part (b) involves the following activities :

1) Accept handshaking signals from the Link Controller Unit and respond.

2) Transfer data from the memory to the Link Controller and vice-versa.

3) Process incoming messages from the link.

4) Process messages transferred from the Host.

5) Respond to Host commands.

All these are done under the control of the 8085A CPU. The transfer of data to and from memory is done by an 8257 DMA controller along with several other support chips like 8212, 74257 etc.

Part (c) is the Host Interface Unit. The Host Interface is a serial link from the host terminal to the IMP. The following jobs are done by this part :

1) Sense serial input from host terminal, convert to parallel bytes and send it to IMP's CPU.
2) Generates handshaking signals for the CPU-host interaction.
3) Receive parallel bytes from the IMP's CPU convert to parallel bytes and send out to host terminal.
4) Convert the unipolar TTL level (+5V, OV) signal to the Bipolar RS-232 standard (+12V, -12V).

The 1st three operations are done by an 8251A USART. For the last operation Motorola 1458, 1459 RS-232 transmitter and receiver are used.

Over and above the three major parts there is also part (d) viz., The Timing and Interrupt Generation Unit. This interacts with all the above three parts. Basically it consists of an 8253 programmable counter, along with TTL support chips like 7493s and 7474s.

It does the following operations :

1) Generates Baudrate for USART
2) Generates Baudrate for Link Controller Unit (i.e. $\overline{TXC}$)
3) Generates clock for data recovery (i.e. 32X CLK)
4) Generates time out for sending EOP character (primary only)

The schematic diagram of the complete Primary/Secondary station is given in Fig. 2.01.

Table 2.01 gives the functional break-up of the IC's used.

2.1 THE PROGRAMMABLE SDLC/HDLC PROTOCOL CONTROLLER INTEL 8273

(a) The 8273 Chip - The 8273 is a synchronous serial I/O device which has been designed to support HDLC and SDLC protocols. Though designed with the INTEL 8080A and 8085 microprocessors in mind, it can be used with almost any microprocessor.

The 8273 device logic can be broadly divided into serial trasmit, serial and supervisor sections.

The serial transmit and receive section, as their names imply, contain the logic which enables serial data to be transmitted or received. The supervisor section contains the logic which allows us to select options under program control and to monitor operations in progress.

We can divide 8273 operation into these four steps :

1) Initially, we must define the overall conditions under which the 8273 is to operate. We do this, by writing command and parameter registers in the supervisor section.

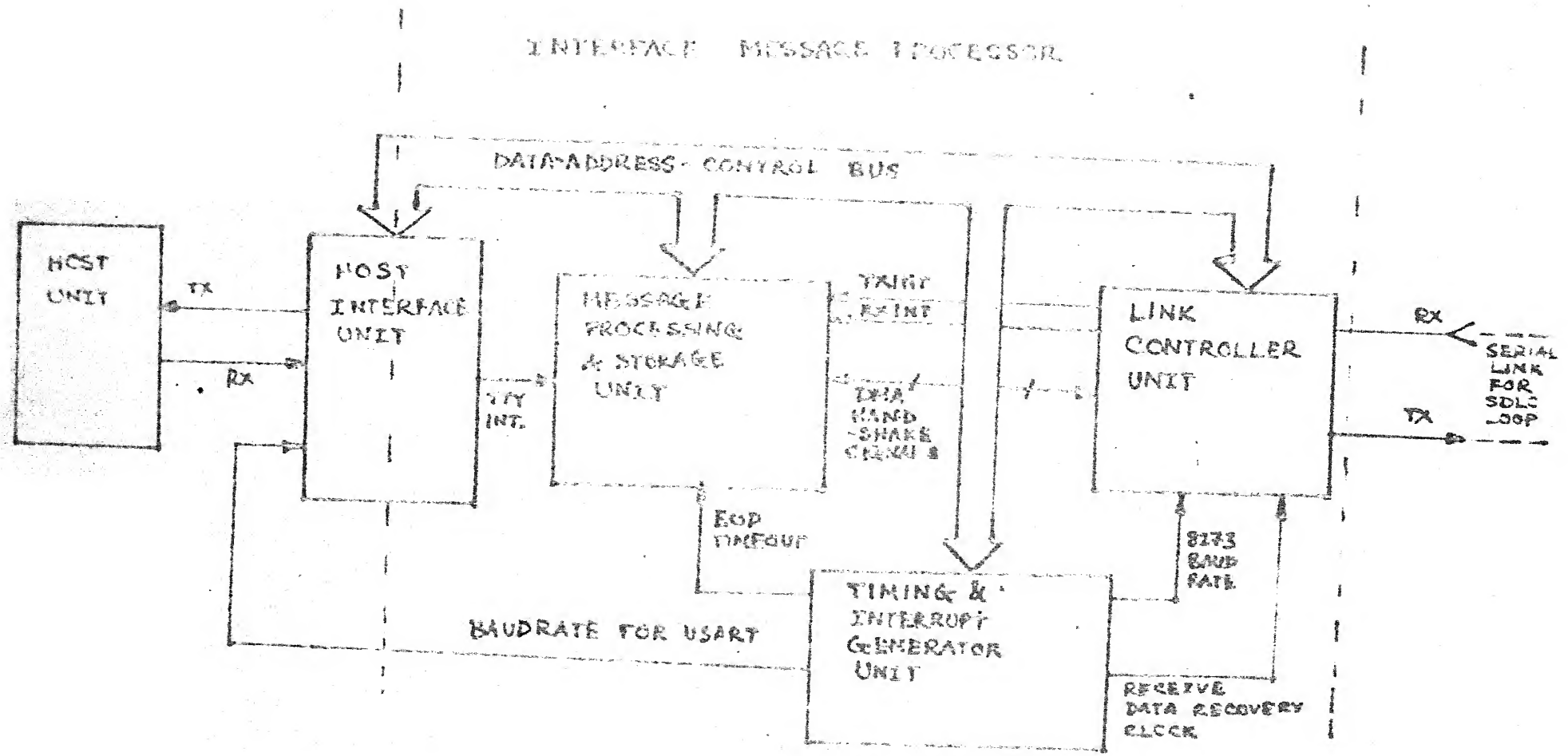


FIG: 2.01 SCHEMATIC DIAGRAM OF A LOOP STATION

List of IC chips and their functions (Ref: Appendix-A, detailed circuit diagram)

8085 : The CPU

8273 : The SDLC protocol controller

8257 : Direct Memory Access Controller for transferring data between 8273 and memory in either direction

8251 : USART - Parallel to serial Asynchronous interface between the IMP and Host terminal

8253 : Programmable timer. Used for the following purposes :

a) Band rate generator for the USART data transfer

b) Band rate generator for the 8273 data transfer

c) Clock for data recovery by DPLL of 8273

d) Time out generator for EOP character passing

8205 : Address decoder for memory chips and peripherals

8212(A) : To provide the interrupt vector RST 1

8212(B) : Address data latch for 8085A

8212(C) : Higher Address latch for 8257

2716 : EPROM

2114 : RAM

contd...

75188 } 75189 }	: For RS-232 compatible interface between IMP and Host 8251
74257	: Control bus multiplexer for 8085A and 8257
74125	: Tristate buffer
7493	: Divide by 16 counter for supporting 8253
7474	: D-F/F for supporting 8253 and generating 1.5 MHz clock for 8253 and 8251

Other SSI's like NAND and OR gates are also used

Table 2.01 List of IC chips and their functions

2) Within the overall conditions defined in Step 1, we will set up transmit logic to transmit a frame, and/or we will set up receive logic to receive a frame. We do this, by writing further commands to the command and parameter registers. Of the supervisor section. It is important to note that even though transmit and receive and receive logic are separate, the same command and parameter registers are used to initialize transmit and receive operations.

3) Data will be transmitted and/or received as specified in Step 2. Data can be transmitted and received simultaneously. Separate external transmitter and receiver clocks can be provided. Alternatively, we can derive a single transmit/receive clock from the received serial data stream.

4) A transmit or receive operation can end successfully or unsuccessfully. The outcome is reported to the TXINTR and RXINTR registers.

The 8273 has been designed to work with interrupt logic and/or direct memory access logic.

Transmit and receive logic always use interrupts to report error conditions, as well as the successful transfer of a data frame. While the data frame is actually being transferred, transmit and receive logic always use their memory access

signals to transfer parallel data to or from the microprocessor system. We can, however, cause interrupt requests to be generated under program control along side DMA requests.

The functional diagram, pin connection and signal assignment of the 8273 are given in Figs. 2.11 and 2.12.

(b) 8273 Command Structure - The 8273 device is controlled via a set of device commands. All commands (with the exception of RESET) write an 8-bit code into the command register. For most commands additional code bytes must be written into the parameter register. Timing is vitally important when we output commands to the 8273 device. To guarantee correct timing we must use the status register when writing into command and parameter registers. The status register bits are shown in Fig. 2.13.

The peculiarity of 8273 device commands is that each command has a counter command. Options that are set by one command are reset by its counter command, through their respective parameter registers. The important commands of 8273 are summarized in Table 2.11.

2.2 INTERRUPT STRUCTURE USED IN THE STATION

From the above discussion it is clear that the transmit and receive logic of 8273 always use interrupts to report the successful transfer of a data frame as well as error conditions. So for this we need two interrupts. Again in the

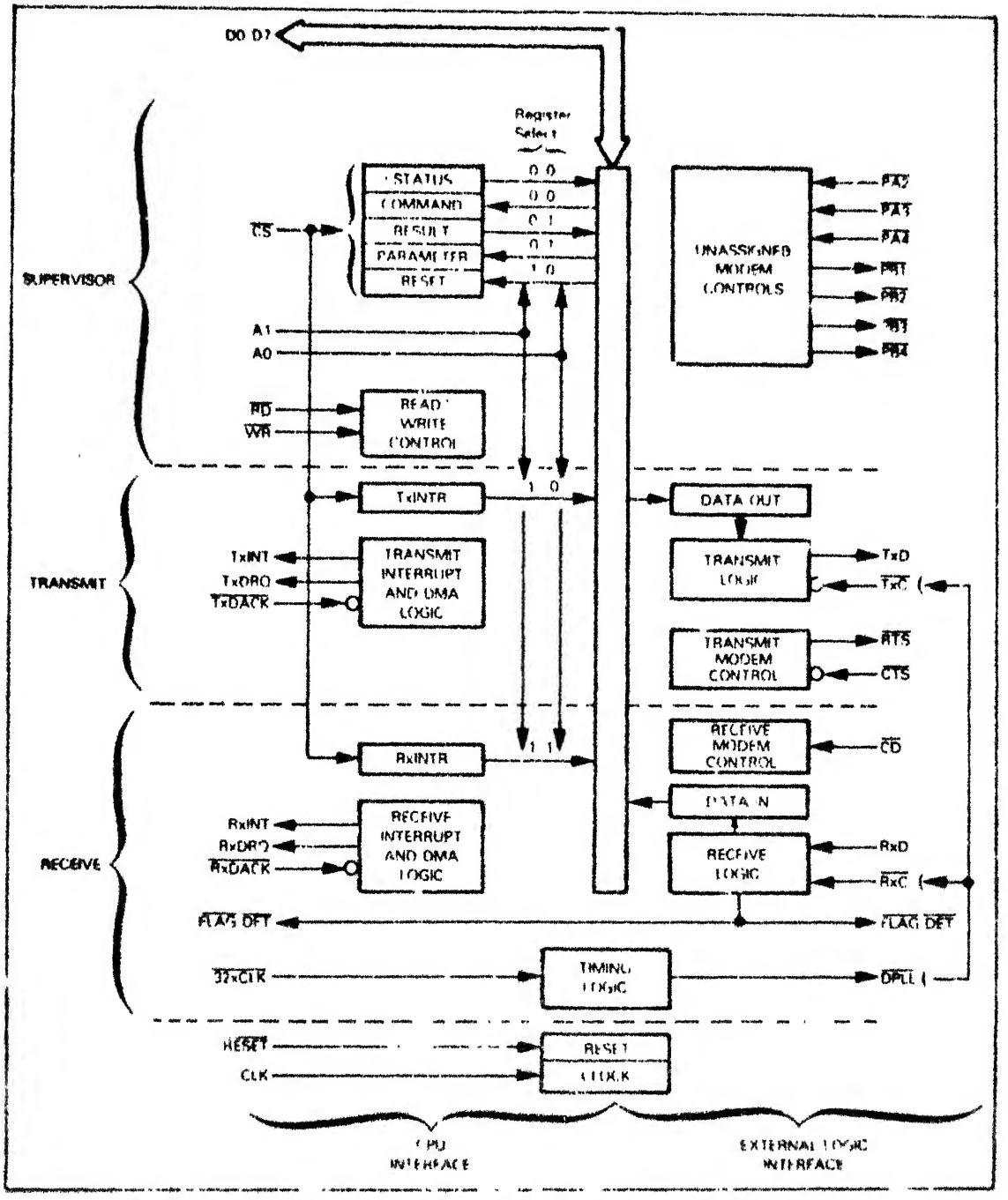
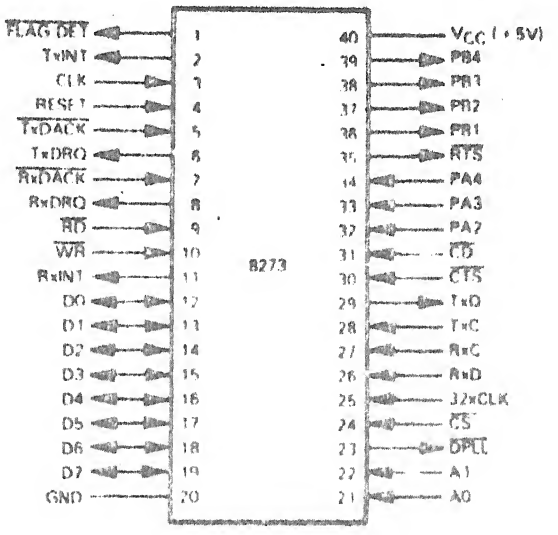


FIG: 2.11 FUNCTIONAL LOGIC OF 8273



PIN NAME	DESCRIPTION	TYPE
D0 - D7	Data Bus	Bidirectional, tristate
A0, A1	Register Select	Input
$\overline{CS}$	Non-DMA device select	Input
$\overline{RD}$	Read control	Input
$\overline{WR}$	Write control	Input
PA2 - PA4	Unassigned inputs	Input
PB1 - PB4	Unassigned outputs	Output
TxD	Serial data output	Output
TxC	Serial data output clock	Input
$\overline{RTS}$	Request to send	Output
$\overline{CTS}$	Clear to send	Input
TxINT	Transmit logic interrupt request	Output
TxDRQ	Transmit logic DMA request	Output
$\overline{TxDACK}$	Transmit logic DMA acknowledge and device select	Input
RxD	Serial data input	Input
RxC	Serial data input clock	Input
$\overline{DC}$	Data Carrier detect	Input
FLAG DET	Flag detect	Output
RxINT	Receive logic interrupt request	Output
RxDRQ	Receive logic DMA request	Output
$\overline{RxDACK}$	Receive logic DMA acknowledge and device select	Input
32xCLK	Asynchronous clock generator	Input
DPLL	Asynchronous clock	Output
CLK	System clock	Input
RESET	System reset	Input
VCC, GND	Power (+5 V) and Ground	

FIG: 2.12 PIN CONNECTIONS & SIGNAL ASSIGNMENT OF 8273

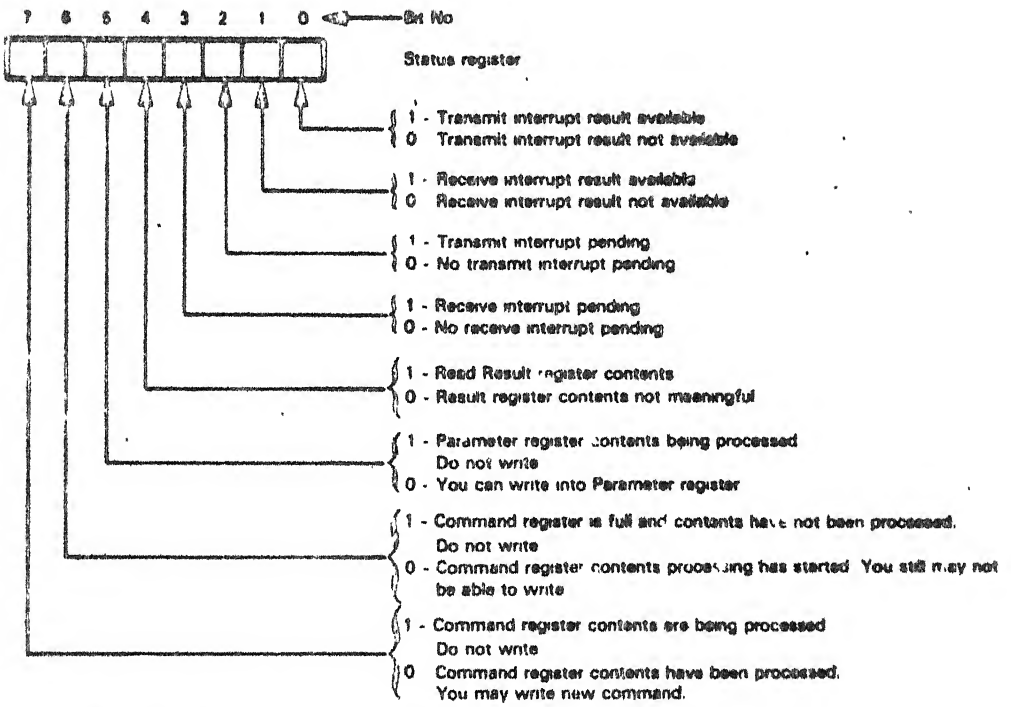


FIG 2.13 STATUS REGISTER BITS

Command Type	Command Name	To Reset Register	To Command Register	To parameter Register	Function
1	2	3	4	5	6
Transmit Logic Control (These commands all generate results in the TXINT register)	Transmit Frame		C8	B(LO) B(HI) (A) (C)	8273 transmits one frame including B data bytes. The A and C bytes are provided as parameters in buffered mode only
	Loop Transmit		CA	B(LO) B(HI) (A) (C)	Transmit a frame of B data bytes in SDLC loop made. The A and C bytes are provided as parameters in buffered mode only. Transmission begins after reset flag is received or if flags are not being received after an EOP(7F16) is received and converted into a flag. At end of loop transmit one bit delay is set(= A430) and idle output occurs (51-parameter bit Ø = 0)

1	2	3	4	5	6
Modem Control	Set Port B		A3	ØØXXXXXX	Modem output signals are set if X = Ø or reset if X = 1 as follows :
	Reset Port B		63	11XXXXXX	$\overline{RTS}$, if $\overline{RTS}$ is reset low, it remains low. If $\overline{RTS}$ is set high, it will be forced low for the duration of each frames transmission. PB1 These outputs are PB2 set or reset but PB3 play no part PB4 in any operations $\overline{Flag\ Detect}$. If this is forced low, flag characters will not be detected. If it is forced high, flags will be identified by a low pulse.
Initia-lization	Set one bit delay		A4	80	8273 is a slave in an SDLC loop. It transmits what it receives.
	Reset one bit delay		C4	7F	Cancel one bit mode
	Interrupt mode		97	Ø1	Use interrupt logic to transfer parallel data.
	DMA		57	FE	Use DMA logic is transfer parallel data.

1	2	3	4	5	6
	Set Operating Mode		91	ØØXXXXXX	Select(using X=1) or deselect (using $\bar{X}$ = 0) operating modes as follows :
Initialization					Transmit flags(X=1) or idle($\bar{X}$=0) when inactive Transmit (X=1) or not preframe sync.character Buffer (X=1) or hold in memory ($\bar{X}$=0) A and C Transmit (X=1) or do not transmit ($\bar{X}$=0) early EOF interrupt Do (X=1) or do not transmit ($\bar{X}$=0 EOP interrupt,assuming SDLC loop mode Select HDLC abort (X=1) or SDLC abort ($\bar{X}$=0)
	Reset Operating Mode		51	11XXXXXX	
	Set Serial I/O mode		AØ	ØØØØØXXX	Select (X=1) or diselect($\bar{X}$=0) Serial I/O options as follows :
	Reset serial I/O mode		6Ø	11111$\bar{X}\bar{X}\bar{X}$	Select NRZI(X=1) or normal positive logic ($\bar{X}$=Ø) Loop back TXC to RXC (X=1) or operate normally ($\bar{X}$=0) Loop back TXD to RXD (X=1) or operate normally ($\bar{X}$=0)

1	2	3	4	5	6
Reset	Reset	Ø1 ØØ			Reset device. This is equivalent to low RESET input
Receive logic control (These commands all generate result in RXINT register)	General Receive		CØ	B(LO) B(HI)	8273 ignores device address and receive logic receivers B data bytes
	Selective Loop Receive		C2	B(LO) B(HI) A1 A2	8273 receive logic operates in SDLC

Table 2.11 Summary of important 8273 Commands

primary station the token generation process is timed out by 8253 programmable counter which necessitates a third interrupt. Moreover, efficient message processing and link data handling requires that the host-imp interaction be also through keyboard interrupt (via 8251A). This ensures minimum busy waiting period for the CPU and maximum CPU utilization. Thus the interrupt hierarchy has to be chalked out carefully. Specifically one must avoid a) Loss of link data b) clashing of interrupts leading to deadlock conditions, c) conflicting update of shared resources (memory and IO) by different interrupt processes and d) uncontrolled nesting leading to stack overflow.

By a proper selection of interrupt hierarchy and suitable enabling and disabling of interrupts, we can arrive at a situation free of the above problems. Working on these criteria, our design has the following interrupt structure.

1) RST 7.5 is used as the Rx. interrupt from the 8273. Since we are dealing with a communication link and data can be received at any time, the highest priority interrupt is used here (we are not using TRAP and we keep it disabled).

2) RST 6.5 is used as the EOP timeout interrupt

3) RST 5.5 is used here as the user interrupt

4) INTR is used for transmit interrupt. The justification of using a low priority interrupt for transmission is that

it takes place at very specific places in the IMP routines and has minimum interaction with other interrupts.

2.3 IO MAP DETAILS

The Table 2.30 describes the details of IO mapped device addresses on functional basis. There is some specific sequence of access for most of the devices which would be useful to note here. The 8257 does not require any sequence in the way its various registers are accessed. However, they are all 16 bit registers. Hence they need to be programmed by two consecutive bytes between which there is a specific sequence namely Lo order byte first, HI order byte next. The 8251 has a peculiarity in the sense that its command register can be in two modes, mode set or command. Unless the mode is set command register does not accept commands. On power on the register can be in any state. 8253 registers are also sequence independent. But double byte programming needs a sequence very much like that of 8257. 8273 registers are however all sequence dependent. The sequence is controlled by monitoring the status word.

2.4 DETAILS OF MEMORY MAP

For our program space we have kept 4K byte of PROM. For data storage, scratch pad and stack we have kept 2K byte of RAM. We have kept 8K byte space for future expansion both in terms of RAM and PROM. Expansion of RAM is specially expected

Device Name	Port Address (Hex)	Read Location Accessed	Write Locations Accessed
1	2	3	4
8257	ØØ	Ch-Ø DMA Address	Ch-Ø DMA Address
	Ø1	Ch-Ø Terminal Count	Ch-Ø Terminal Count
	Ø2	Ch-1 DMA Address	Ch-1 DMA Address
	Ø3	Ch-1 Terminal Count	Ch-1 Terminal Count
	Ø8	Status Register	Mode set register
8251A	10	Data out Register	Data in Register
	11	Status Register	Command Register
8253	2Ø	Counter Ø	Counter Ø
	21	Counter 1	Counter 1
	22	Counter 2	Counter 2
	23	No operation 3 state	Mode word

1	2	3	4
8273	30	Status Register	Command Register
	31	Result Register	Parameter Register
	32	Transmit Interrupt Register	Reset Register
	33	Receive Interrupt Register	None

Table 2.30 Details of I/O Map

N.B: 8273 is only partially I/O mapped. Its 'Data In' and 'Data Out' registers are accessed by DMA handshake signals $\overline{\text{RXDACK}}$ and $\overline{\text{TXDACK}}$ respectively.

for dealing with larger messages, and having such additional features like data editing or file system in future. Table 2.40 gives the detail of Memory Map.

2.5 SOME SPECIAL HARDWARE DESIGN FEATURES

Refer : to the Detailed Circuit Diagram in Appendix-A.

i) The secondary hardware is a subset of the primary station hardware. The additional requirement for the primary are the following :

a) Loop Clock - As the loop controller, the primary has to generate the source clock to drive data around the loop. All the loop terminals(viz.secondaries) capture the loop clock by their internal Digital Phase Locked Loop (DPLL) and both transmits and receives by the same generated clock. However, both primary and secondary must be provided with the 32xCLKinput for recovering the loop clock from - received data. Fig. 2.51 depicts the related schematic hardware diagram.

b) Time out clock for token generation - For the EOP-time out interrupt generation the Primary needs a counter. Counter 2 of the 8253 has been used for this purpose with the counter programmed in mode 0 (software triggered interrupt). However, in the Main program it is convenient to have control over the actual counting period of the timer through its gate. This is achieved by connecting the SOD output of 8085A to the

Address Range	Length (bytes)	Device mapped
ØØØØH-07FFH	2K	2716 (PROM 1)
Ø8ØØH-ØFFFH	2K	2716 (PROM 2)
1ØØØH-13FFH	1K	2114 (RAM1 and RAM 2)
14ØØH-17FFH	1K	Unconnected (folds over 1ØØØ-13FFH)
18ØØH-18FFH	1K	2114 (RAM 3 and RAM 4)
1CØØH-1FFFH	1K	Unconnected (folds over 18ØØ-18BFH)
2ØØØH-3FFFH	8K	For future expansion

Table 2.40 Details of memory map

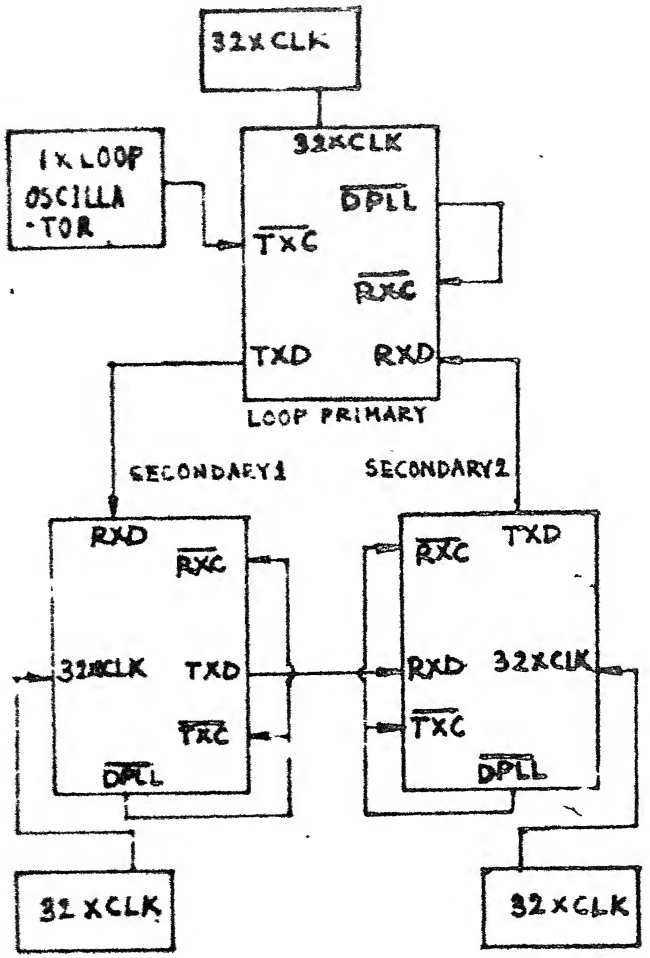
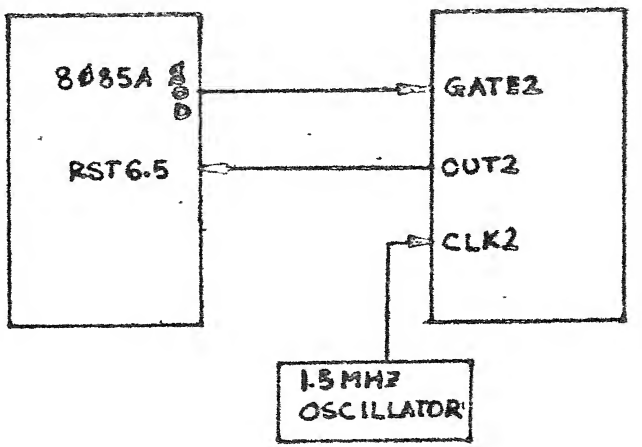


FIG: 2.51 THE LOOP CLK CONNECTION IN PRIMARY AND SECONDARY.



SOD : ENABLES/DISABLES TIMEOUT COUNT

OUT2 : GENERATES TIMEOUT INTERRUPT

FIG: 2.52 TIME OUT SCHEME FOR TOKEN GENERATION IN PRIMARY

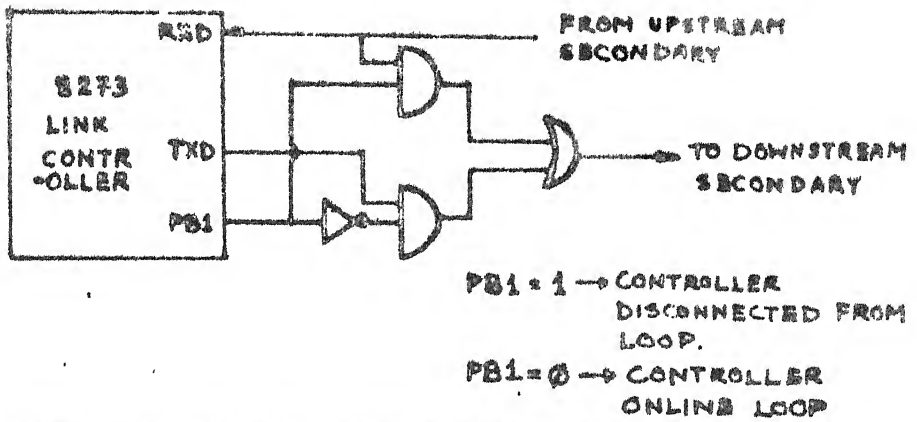


FIG: 2.53 ONLINE / DISCONNECT FECILITY

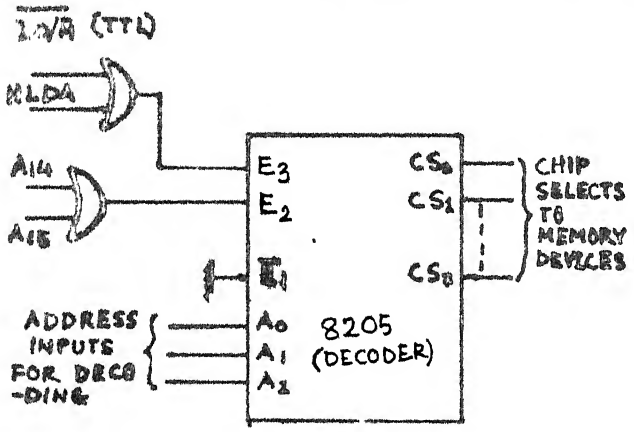


FIG: 2.54 ACTIVATION OF RAM SELECTION LOGIC DURING DMA.

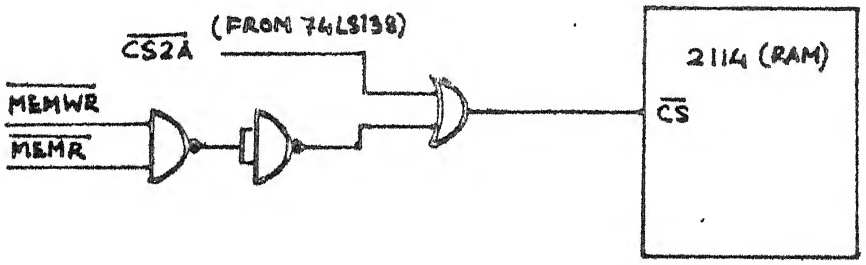


FIG: 2.55 CIRCUIT FOR CONFLICT FREE SELECTION OF RAM.

gate 2. With this provision, the timeout counting can be freezed as and when necessary. Fig. 2.52 depicts the schematic arrangement.

[N.B. See Appendix-B for the modification needed to make the station programmable both as secondary and primary].

ii) Each secondary station is provided with the facility of getting itself online or disconnected under program control. This is done by using one of the Port B modem control signals (PB_0) to enable or disable the Txmit logic and connect its input (RXD) to output externally through four 7400 NAND gates. The circuit schematic is shown in Fig. 2.53.

iii) During DMA operations the I/O devices including 8257 has to be deselected. This is done by the standard way of using AEN as one of the enable inputs to the I/O chip select decoder. But we also have to ensure that the RAM selection logic remains enabled for DNA to take place. This is necessary as one of the enables for Memory decoder is $IO/\overline{M}$, which becomes tristated while DMA is in operation and fails to generate proper selection logic. Consequently we have used HLDA to override $IO/\overline{M}$ in memory chip select enable while DMA is active. The circuit schematic is given in Fig. 2.54.

iv) The memory decoder circuit takes into account another peculiarity of the 2114A (RAM) chip itself. In the RAM 'write' cycle, if the $\overline{WE}$ ($\overline{\text{Write Enable}}$) remains high at the low going edge of the $\overline{CS}$, then the 2114A starts driving data through its

output (D_{out}) buffer. This, of course, leads to bus conflic with the incoming data on the system bus. To avoid, this, the $\overline{WE}$ has to simultaneously become low with $\overline{CS}$. Then the D lines becomes Hi-impedance and data can safely be written into the RAM. The circuit schematic for this arrangement us NAND gate is shown in Fig. 2.55.

v) As a precautionary measure, wherever in the circuit MOS output is required to drive more than one TTL input it has b buffered. For this purpose the 74125 tristate buffer as wel as ordinary TTL inverters have been used, depending on the particular application.

As can be seen from the description given in the previous sections, apart from some special features, the hardware design is an almost standard one. The software needed to make this hardware functional is described in the next chapter.

CHAPTER 3

DESIGN OF SOFTWARE FOR THE IMP

3.0 BASIC DESIGN REQUIREMENT

Before proceeding to describe the actual implementation of the software, we shall briefly narrate the following points which are considered as the basic system requirement in the design and which provides the guiding philosophy of the software actually implemented. They are as follows :

1) Our aim is to develop the operating system for the Interface Message Processor Unit. We assume the availability of a Host on one side and the serial link on the other.
2) Our design is restricted to the data link and Network layer of the loop net. Under implementation. The physical layer, namely the communication channel is assumed to exist. However, on the user side, some user interaction has been necessitated by the fact that, what we have used here as host is basically a dumb terminal.
3) All data link - Network layer activities have to be absolutely transparent to the user in our system.
4) The software to be written must be absolutely modular to make it flexible and easily updatable.

5) The software should be perfectly general. It should ensure that introduction or withdrawal of any number of nodes (subject to the maximum) does not hamper the loop operation.

6) A reasonable number of supervisory features have to be added in the loop primary station to guarantee smooth operation and fair access for all nodes.

7) Optimization of servicing delay and effective link utilization have to be taken into account, however, under the constraints of the protocol used.

3.1 DATA STRUCTURE FOR THE IMP

The data structure for the primary and the secondary station are made identical as far as possible. This data structure can be functionally categorized into the following :

a) Receive data structure
b) Transmit data structure
c) Message processing data structure
d) Acknowledgement data structure
e) Supervisory data structure
f) Command processing data structure

All these data structures are operated upon by specific routines and interact heavily with one another.

a) Receive Data Structure

The receive data structure relates to variables required for manipulation of the Receive Buffer. The general condition of the Rx. buffer is one where messages from multiple sources, of variable type, length and destination reside together. So the basic requirement is a queue of the Rx. message headers. Fig. 3.11 shows this queue which starts from the address RDS. It is a multiple field queue with Result byte, Buffer length Lo, Buffer Length HI, Frame Address Frame code, Start Address Lo and Start Address HI being the seven fields of an element of the queue. Every such element will be known as Per Message Data Structure (PMDS). So the header queue is a queue of individual $PMDS_i$ elements. The depth of the queue designed is 8 elements (i.e. 8x7 = 56 bytes). If more than eight messages are received before a message gets processed (and queue updated) the queue will wrap around itself (cyclically) and header of the 1st of these messages will get overwritten by that of the new message. In that case the first message would be lost.

Accessing of the message queue is done as shown in Fig. 3.11. Here, we have a queue pair RTABL and RTABH to point to the $PMDS_i$ Start Address. This is a standard way of accessing multiple field queues. Therefore ($RTABL_i$) gives the start Address of $PMDS_i$. The Top and Bottom of the queues RTAB

PMDS:

RESULT BYTE
BUFFER LENGTH LO
BUFFER LENGTH HI
FRAME ADDRESS
FRAME CODE
START ADDRESS LO
START ADDRESS HI

TCRTL

TOP OF QUEUE
RTABL

TCRTH

TOP OF QUEUE
RTABH

BCRTL

BOTTOM OF QUEUE
RTABL

BCRTH

BOTTOM OF QUEUE
RTABH

RTABL

POINTS TO $PMDS_1$ (LO)
POINTS TO $PMDS_2$ (LO)
⋮
POINTS TO $PMDS_n$ (LO)

RTABH

POINTS TO $PMDS_1$ (HI)
POINTS TO $PMDS_2$ (HI)
⋮
POINTS TO $PMDS_n$ (HI)

FIG: 3.11 RECEIVE DATA STRUCTURE

and RTABH are designated TCRTL, BCRTL and TCRTH, BCRTH respectively. They are both of double byte length.

This structure ensures that knowledge of just two variables is sufficient to process a message either at the Top (through TCRTL) or Bottom (through BCRTL) of the message Buffer. We examplify this below.

Suppose we have 2 messages currently residing in the Rx. buffer, a 3rd message has come and we have to dump it (with its header pointing to it) in the Rx. buffer. So we need to access the 'Bottom' of the Rx. Buffer. We first access BCRTL. (BCRTL, BCRTL+1) will give $RTABL_3$ in this case. Now we access BCRTH. (BCRTH, BCRTH+1) will give $RTABH_3$. ($RTABL_3$, $RTABH_3$) will give $PMDS_3$. $PMDS_3$ is the Start Address of the header for the message just received. So we would now go on storing the message header from $PMDS_3$ to $PMDS_3 + 6$. In doing that the Start Address of the actual message would be stored in $PMDS_3$+5, $PMDS_3$+6. So any access to the actual message in future would be through $PMDS_3$+5, $PMDS_3$+6.

Similarly in case we want to process a message we access the PMDS through the top TCRTL, TCRTH of the queue pair RTABL, RTABH with a 2-level indirection.

b) Transmit Data Structure

The transmit Data Structure is exactly similar inform to the Rx. Data Structure. However, unlike the case of Receive

PMDS

RESULT BYTE
BUFFER LENGTH LO
BUFFER LENGTH HI
FRAME ADDRESS
FRAME CODE
START ADDRESS LO
START ADDRESS HI

TNEWL

TOP OF QUEUE TTABL

TNEWH

TOP OF QUEUE TTABH

BNEWL

BOTTOM OF QUEUE
TTABL

BNEWH

BOTTOM OF QUEUE
TTABH

TTABL

POINTS TO PMDS1 (LO)
POINTS TO PMDS2 (LO)
POINTS TO PMDSn (LO)

TTABH

POINTS TO PMDS1 (HI)
POINTS TO PMDS2 (HI)
POINTS TO PMDSn (HI)

FIG: 3.12 TRANSMIT DATA STRUCTURE

queue, not all transmissions take place through the transmit queue. Acknowledgement and Polling frames (in case of Primary only) are transmitted independently of the transmit queue. Hence, the transmit queue is used to transmit message and command frames only. Fig. 3.12 depicts the Transmit Data Structure.

c) Message Processing Data Structure

The Message Processing Data Structure relates to the variables required for actual processing of information, command and acknowledgement frames. This data structure forms the bridge for transfer of information between the Link side Routines on the one hand and the User side routines on the other. Fig. 3.13 depicts the message processing data structure.

The variables CINDX, CAD, CCD, CBLL, CBLL+1, CSADL, CSADL+1 stores the header information of the message currently being processed. They thus save processing time needed to indirectly access the message while it is being processed. CINDX contains the RTABL Address (i.e., Start Address of its PMDS) of the message. CAD points to the transmission address (sender, in case of secondary, receiver in case of Primary) of the frame. CCD contains the code or control word of the frame being processed. CBLL, CBLL+1 contain the buffer length occupied by the current message. CSADL, CSADL+1 points to the actual start address of the current message in the Rx. buffer.

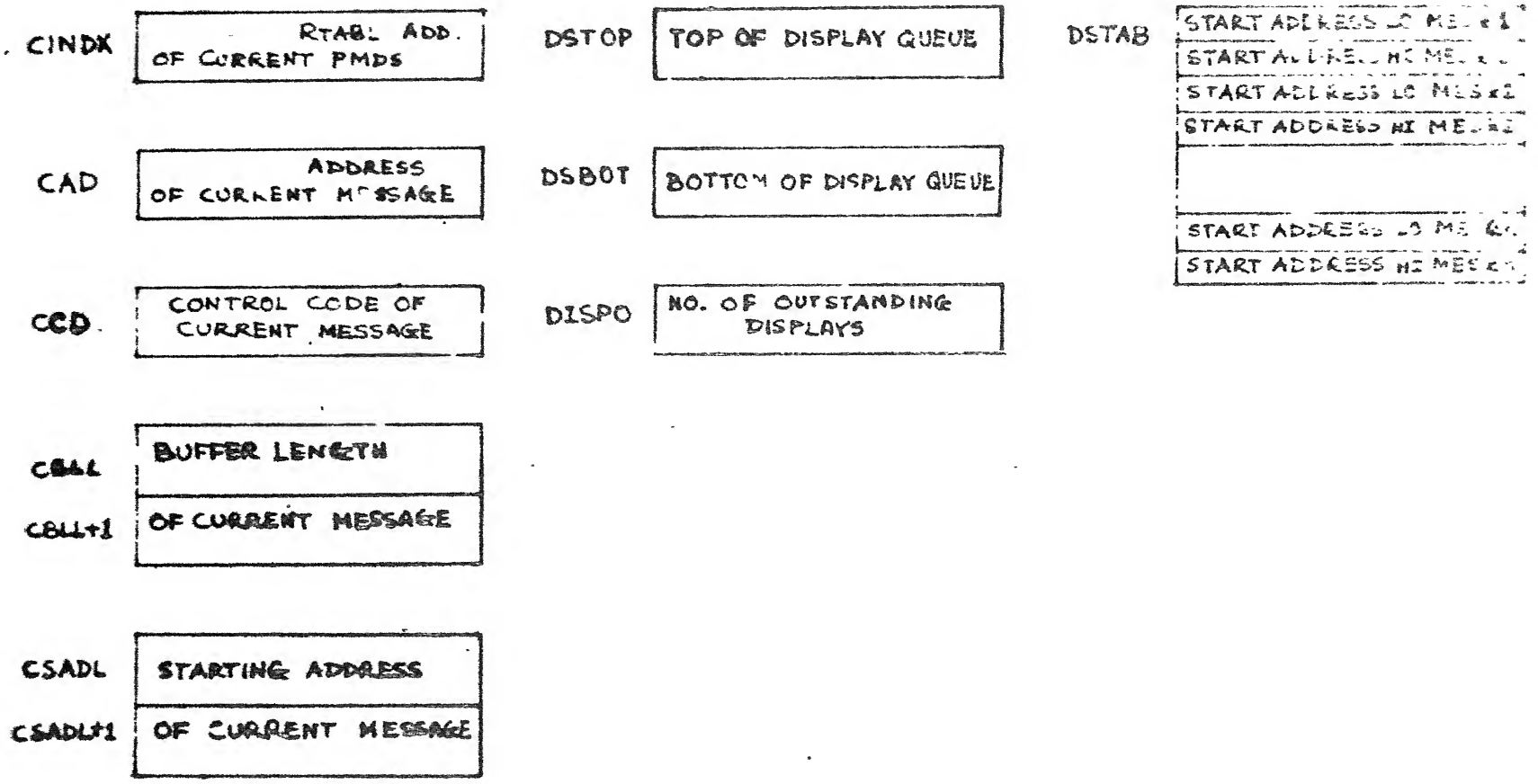


FIG: 8.13 MESSAGE PROCESSING DATA STRUCTURE

Thus all these quantities correspond exactly with those used for the Rx. data structure variables. The variables DSTAB, DSTOP, DSBOT and DISPO relates to the displaying of information frames. They thus form a separate substructure within the Message Processing Data Structure. DSTAB is the starting address of the display queue. This queue is actually the queue of start address of messages to be displayed. DSTOP and DSBOT are respectively the top and bottom of the display queue. DISPO contains the number of outstanding messages for display.

d) Acknowledgement Data Structure

As we have stated already, the acknowledgement frames are transmitted independently of the Txmit queue. Moreover, in this implementation we have done without an Acknowledgement queue structure. This is possible by immediately transmitting a pending Acknowledgement (if any) after a message is processed. The processing of acknowledgement is over before we take up the next message for processing. This ensures that there would not be any accumulation of acknowledgement frames.

Fig. 3.14 shows the Acknowledgement Data Structure. Referring to the figure, it is evident that the variables AKAD, AKCD, AKBLL, AKBLL+1 and AKBT, AKBT+1 correspond to those of the header for the Acknowledgement. We should however mention two points of difference here between the Primary and the Secondary structure.

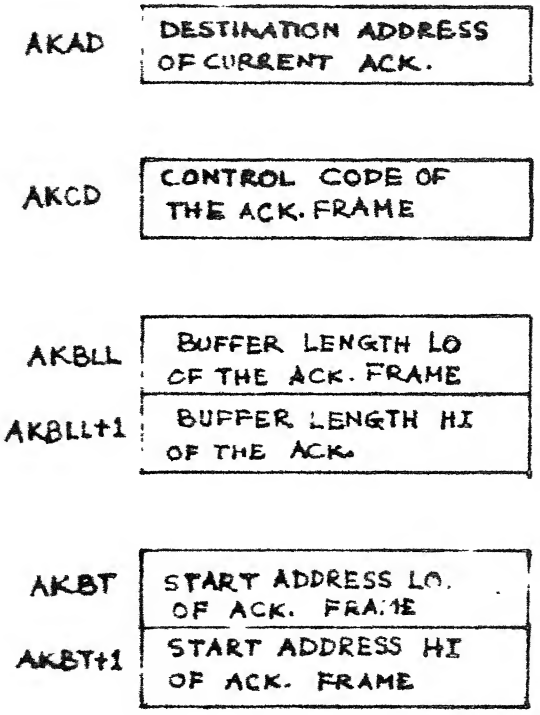


FIG: 3.14 ACK. DATA STRUCTURE IN PRIMARY

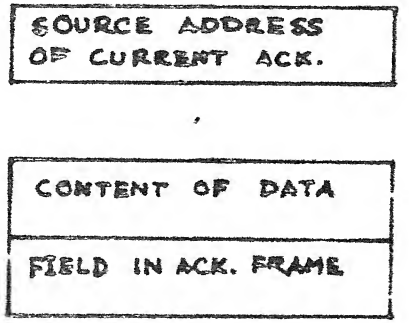


FIG: 3.15 THE DIFFERENCE IN ACK. DATA STRUCTURE OF THE SECONDARY FROM THAT OF THE PRIMARY.

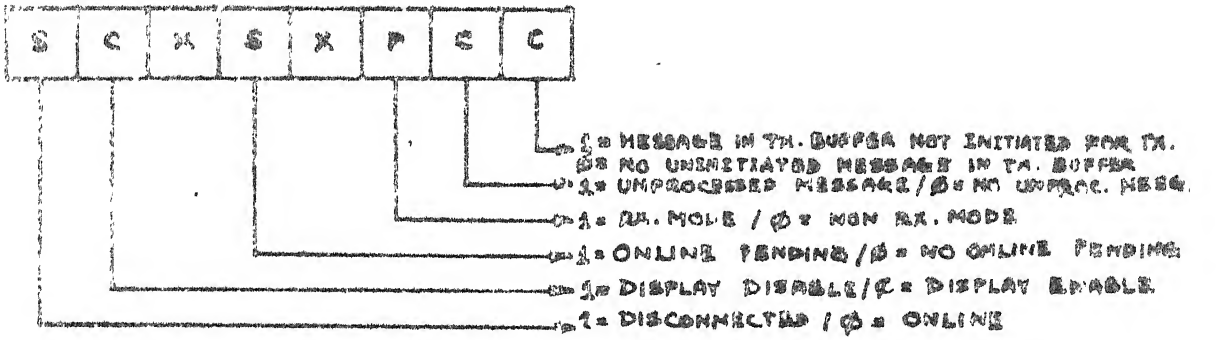
In case of Primary, AKAD refers to the Destination Address of the Acknowledgement frame, whereas, in case of secondary it refers to the source (i.e., the secondary) address. Again, AKBT in case of secondary directly contains the data of Ack. frame (which is atmost single byte). But in case of Primary, the AKBT variable $pt_{.s}$ to the start address of the ack. frame data. This is necessitated by the processing of secondary-to-secondary information in the Primary. The Primary inserts the SSI frame as a dummy Ack frame, thus needing multibyte transmission on acknowledgement. This would be explained in detail in the section on Link side software.

e) Supervisory Data Structure

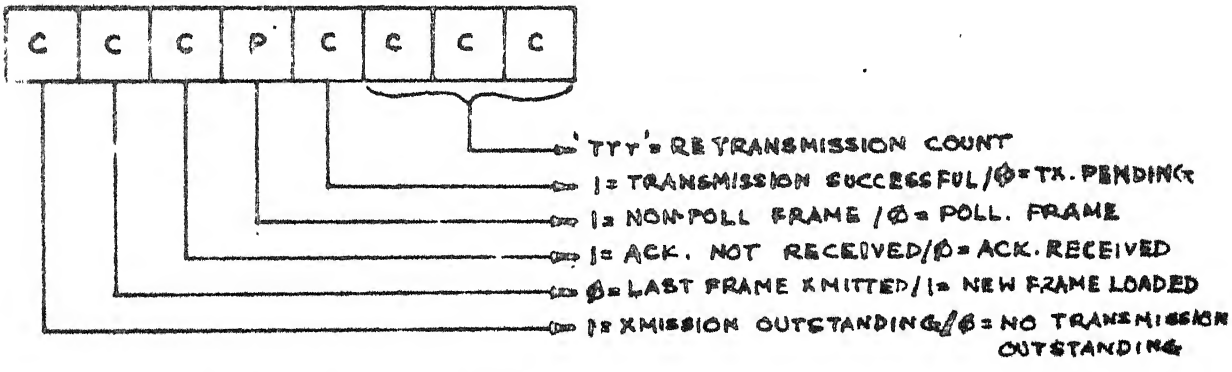
The supervisory data structure relates to variables which are exclusively used for decision making purpose. Hence its utility is not restricted to any specific set of routines. Rather, it is used in determining the future path of execution of the program by controlling or monitoring the program status at every point through its variables.

It is to be noted that it is in this part of the data structure that the variation between Primary and Secondary is quite significant. This reflects the difference in the program flow between the Primary and the Secondary stations at various points.

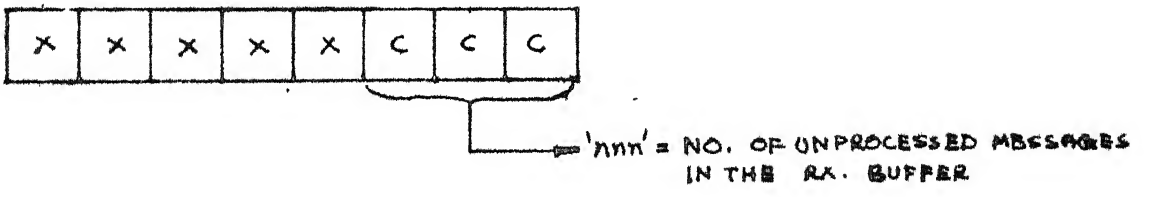
SOPS (STATE OF PROCESSING STATION)



SOBT (STATE OF TRANSMIT BUFFER)



SOBR (STATE OF RECEIVE BUFFER)



[N.B. C = COMMON TO BOTH PRIMARY & SECONDARY
S = SECONDARY ONLY
P PRIMARY ONLY
A = DON'T CARE]

FIG: 3.16 SUPERVISORY DATA STRUCTURE

The Start Address and Terminal Count bytes SALT, TCLT, SALR, TCLR etc. do not fall into the supervisory category. But these variables are accessed or updated by almost all the routines associated with the transmission and reception of messages. Thus their generality makes it appropriate to include them here.

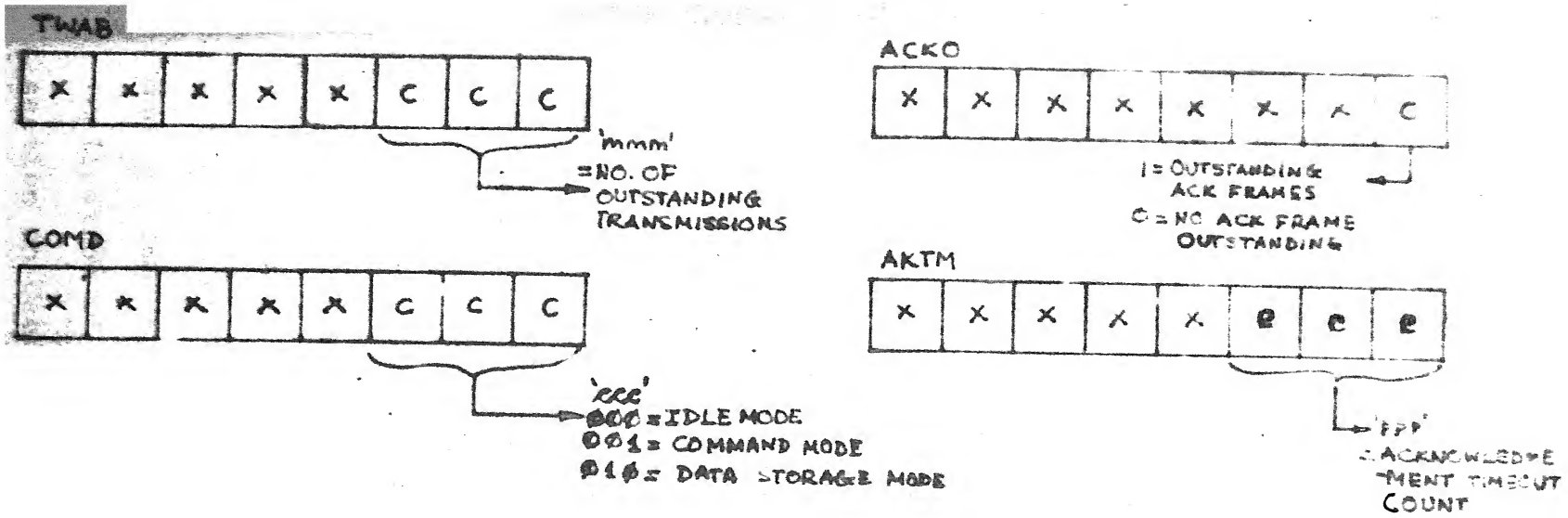
Fig. 3.16 shows the supervisory data structure in details.

SOPS gives the different status bits related to the functioning of the whole station. For example, Bit 7 is set when a secondary station is disconnected from the loop, it is reset when the secondary station is made online. Bit 7 thus controls the on/off line response of the secondary station.

Note, the notations used to designate the characteristics of a bit in Fig. 3.16. P indicates that the bit is meaningful in Primary only and is don't care in secondary. S indicates that the bit is meaningful in secondary operation and is a don't care in Primary. C indicates that the bit is meaningful in both. X indicates that the bit is don't care in both. Small letters (e.g., 'rrr', 'ccc') indicates counter values.

Thus bit 7 of SOPS is relevant to Secondary only, whereas Bit 6 is relevant to both.

The various bits of SOBT gives information related to the transmission of a frame. For example, Bit 7 of SOBT is set whe



SALR	START ADDRESS OF RECEIVE BUFFER	CSAR	CURRENT START ADDRESS FOR RX. DMA
TCLR	TERMINAL COUNT FOR RECEIVE BUFFER	NBLR	NO. OF BYTES CURRENTLY LEFT IN RX. BUFFER
SALT	START ADDRESS OF TRANSMIT BUFFER	CSAT	CURRENT START ADDRESS FOR TX. DMA
TCLT	TERMINAL COUNT FOR TRANSMIT BUFFER	NBLT	NO. OF BYTES CURRENTLY LEFT IN TX. BUFFER

FIG: 3.16 [CONTD.]

a frame is initiated for xmission. It is reset after the frame is successfully transmitted. Similarly, bits 0,1,2 constitute a counter which counts the number of EOP characters received after an unacknowledged xmission.

SOBR is a counter to count the number of unprocessed messages. It is incremented after a frame is received, and decremented after a frame is processed.

TWAB is another counter which counts the number of outstanding messages for transmission.

ACKO indicates the presence or absence of an outstanding ACK. frame.

AKTM counts the ACK. timeout count which is basically the number of retransmissions of a message frame.

SALR, SALR+1, TCLR, TCLR+1, SALT, SALT+1 and TCLT,TCLT+1 are variables associated with DMA initialization. CSAR, CSAR+1, NBLR, NBLR+1, CSAT, CSAT+1 and NBLT, NBLT+1 are corresponding variables associated with the current message.

COMD is a byte associated with the TTY interrupt service routine. It is used to determine the TTY bit response in various cases. If the count value = 000, it indicates idle mode. If count = 001, that indicates command processing mode and if count = 010, it indicates Data Storage mode. The implication of these different modes would be explained in the section on User side software.

f) Command Processing Data Structure

This data structure directly relates to the user utility routines or commands. Fig. 3.17 shows this data structure.

STL is a buffer of 20 bytes used for storing the characters of a command. STP is the pointer to this buffer.

DACL is a byte for temporary storage of the destination address attached as parameter to a command. CODE is a byte for temporary storage of the control code of a command.

TIMER, COUNT and CCNT are counter variables associated with the EOP timeout process.

3.2 MEMORY ORGANIZATION IN THE IMP

For the effective utilization of memory and to assign different tasks to different portions of it, the RAM space of 2K byte has been broken down into different functional areas. The two broad areas are 1) Program support area 2) User area.

The program support area basically constitutes that area of the RAM which is used as scratch pad during program execution and also the system stack. Together, these two parts generate the image of the program at any instant.

The user area constitutes the two main buffers, Rx Buffer and Tx buffer. These two buffers contain data for the consumption of the user only and whatever is written in this area does not directly affect program execution.

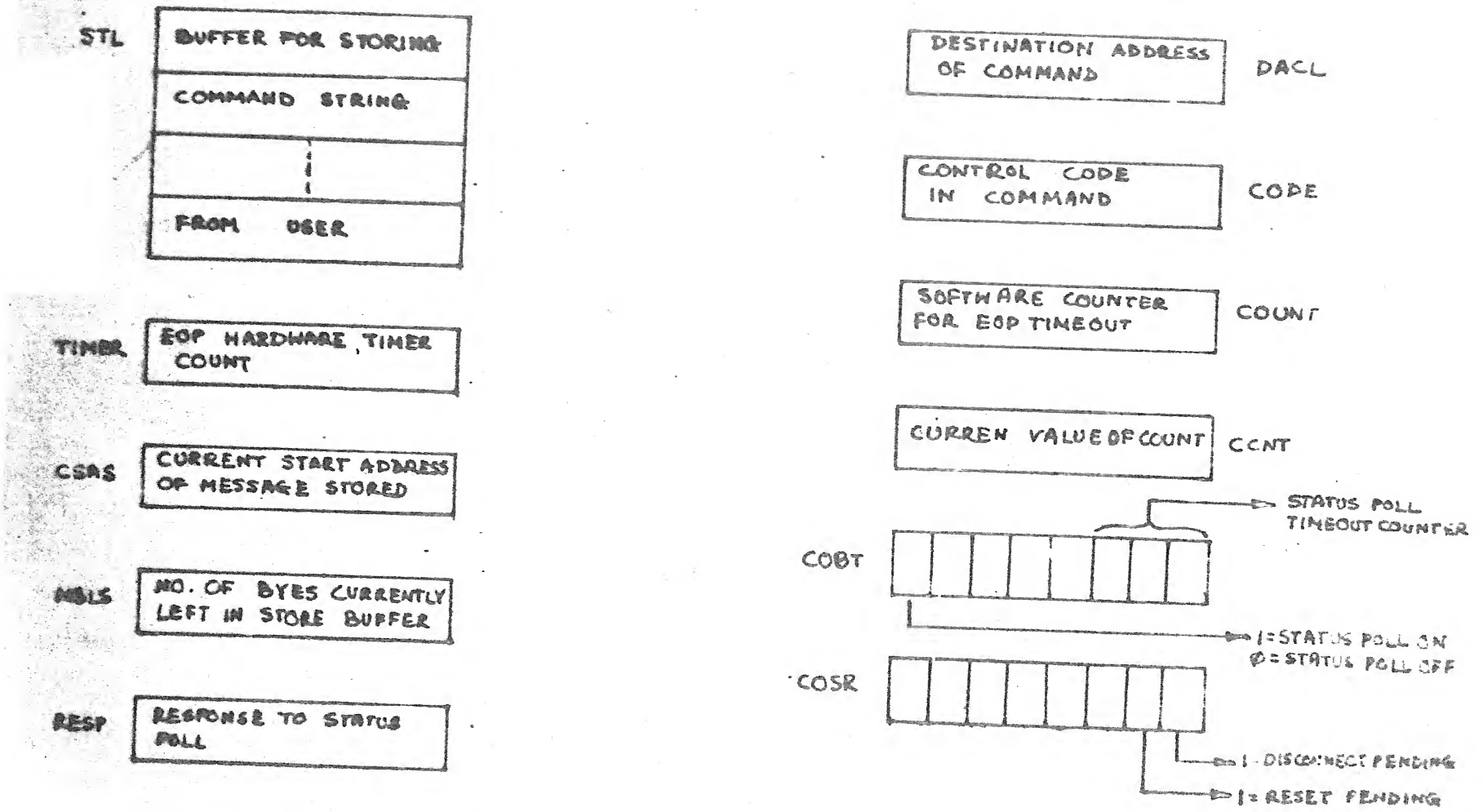


FIG: 3.17 COMMAND PROCESSING DATA STRUCTURE

Sl.No.	RAM SPACE (Hex)	Used as
1.	1ØØØ - 1ØFF	SCRATCH PAD MEMORY
2.	11ØØ - 13FF	TRANSMIT BUFFER
3.	18ØØ - 1B9Ø	RECEIVE BUFFER
4.	1B9F - 1BFF	PROGRAM STACK

Table 3.20 The break up of RAM space

Table 3.20 shows the actual break up of memory space in details.

3.3 PROCESS MANAGEMENT ON SHARED RESOURCES

Before we go into the actual details of the software we will briefly describe some of the problems that occur when multiple processes share common resources (memory and IO). The three main factors that have to be taken into account are 1) Mutual Exclusion, 2) Prevention of Deadlock and 3) Prevention of starvation.

When two or more concurrent processes share a common resource we have to ensure that they exclude each other in time. This exclusion in time implies that until one of the processes finishes operating on the resource, the other cannot start operating on it.

Deadlock occurs when two interdependant processes each wait for the other's execution and as a result none can start executing.

Starvation problem occurs when the execution of one process eliminates the possibility of execution of the other.

All the three problems occur because of disorderly updating of shared resources by concurrent processes. The prevention lies in the creation of, what is known as critical regions which are guarded segments of program which controls the access to that region by some specific methods. The existence of critical

region ensures that the update of common resources is done in an orderly manner.

In our Assembly language level program we have implemented critical regions by two ways. Firstly we have made use of disable interrupt, enable interrupt facilities to give selective access to processes within the guarded regions. These are implemented for interrupt processes. For non-interrupt processes we have used the bit-lock technique. Setting and resetting of certain bits of a variable are done to control the actual sequence of operation. It is needless to say, that manipulation of these decision bits are done outside the critical region.

3.4 SYSTEM INITIALIZATION

The system devices are initialized in the following manner :

1) Initialization of 8251

   a) Mode instruction : OEE_H
      i) Asynchronous with 2 stop bits
      ii) No parity
      iii) Band rate Asyn X 16
      iv) Character length 8 bits
   b) Command Instruction : 15H
      i) Error Reset
      ii) Normal Operation
      iii) Receive Enable
      iv) Transmit Enable

2) Initialization of 8257

a) Mode Set Register : 43H

i) TC stop

ii) Normal write

iii) Fixed priority

iv) Enable channels 0 and 1

b) Address and Terminal count of ch 0

i) Start Address of ch 0 = 1100H

ii) Transmit DMA

iii) Terminal count = 02FFH

c) Address and Terminal Count of ch 1

i) Start Address of ch 0 = 1800H

ii) Receive DMA

iii) Terminal count = 0390H

3) Initialization of 8253

a) Counter 0 Mode word : 36H [Baudrate for 8251]

i) Select counter 0

ii) Read/load least significant byte first, then most significant byte

iii) Mode 2

iv) Binary counter 16 bits

b) Counter 0 count : 0028H

c) Counter 1 mode word: 76H [Baud rate for 8273]

i) Select counter 1

ii) Read/load least significant byte first, then most significant byte

iii) Mode 2

iv) Binary counter 16 bits

d) Counter 1 count: 0005H

e) Counter 2 mode word: OBOH [EOP timeout in Primary]

i) Select counter 2

ii) Read/load least significant byte first, then most significant byte

iii) Mode 0

iv) Binary counter 16 bits

4) Initialization of 8273

I. For the Primary Station

a) Set operating mode

91H : Command register

17H : Parameter register

i) Issue EOP interrupt

ii) Buffered mode

iii) Transmit Flag when idle

iv) Transmit preframe syn. character

b) Set serial I/O mode

AØH: Command Register

01H: Parameter Register

i) Select NRZI

ii) Normal Operation

c) Select DMA mode

57H : Command Register

OFEH: Parameter Register

d) Set PORT B

Command Register : 0A3H

Parameter Register:02H

i) Make PB1 Low

II. For the Secondary Station

a) Set Operating Mode

Command Register : 91H

Parameter Register:04H

i) Buffered Mode

ii) No EOP interrupt

iii) SDLC Abort

b) Set Serial I/O Mode

Command Register : 0A0H

Parameter Register: 01H

i) NRZI select

ii) Normal Operation

c) Set One bit Delay Modes

Command Register OA4H

Parameter Register : 80H

d) Set DMA Mode

Command Register : 57H

Parameter Register: OFEH

e) Reset Port B

Command Register : 63H

Parameter Register: OFDH

i) Reset PB1 High.

3.5 DEVELOPMENT OF PRIMARY SOFTWARE

Let us see the program logic of the Primary Station : If a Primary is idle it should continuously go on polling the secondaries at regular intervals. If it has a frame to transmit then it would wait till the last poll character has travelled through the loop and has been received back by it. Then it would transmit the pending frame. While it is polling secondaries it should wait for the secondary responses to come back to it. On receiving the EOP character back it should start processing the received messages if any. If an information frame is received then it would generate an ack. frame and send it. Further it would update a display queue to display the data received to the user. Similarly if a command frame is received it would process that command and send a response

frame to the sender or send an indication to the host, as the case may be. Further, whenever an user types in something it would respond to the TTY interrupt. On getting a complete frame from the user, it would start processing it. Moreover, on a command generated by the user it should respond to the command and service it.

The primary software can thus bepartitioned into four divisions :

a) Link side routines
b) Message Processing Routines
c) User Side Routines
d) Main Decision Routine

a) The Link side routines handle all the activities on the link. They comprise of the following main routines :

i) Frame Transmit Routine
ii) Rx. Interrupt Routine
iii) EOP Interrupt Routine
iv) Ack. timeout Routine
v) Secondary Polling on EOP Timeout Routine
vi) Transmit Interrupt Routine
vii) Acknowledgement Transmit Routine.

i) Frame Transmit Routine - This routine is invoked from the Main Decision Routine, when there is an outstanding message in the Tx. Buffer which is initiated for xmit, and the acknowledgement to the last frame transmitted has been received. It

accesses the top of the Tx. header queue and initializes the 8257 and 8273 with the header of the top message for transmission. It manipulates SOBT to indicate that an ACK. is to be received and update Tx. Data Structure. It returns to the Main Routine. Fig. 3.5a.1 gives the flow chart of this routine.

ii) Rx. Interrupt Routine - This interrupt routine gets invoked whenever a whole frame is received by the Link Controller unit, successfully, or an EOP character is received or to report any error condition in the receiver. It initiates the header of the Rx. frame into the Rx. buffer header queue, and update a flag to indicate an unprocessed message. It updates Recave Data Structure and returns to the interrupted routine. Fig. 3.5a.2 shows the flow chart of the Rx. interrupt Routine.

iii EOP Interrupt Routine - This subroutine is called from the Rx. Interrupt Routine whenever an EOP character is Received. It increments the Ack. timeout counter if no ack. is received on the last transmitted frame. If the counter threshold is exceeded it calls ack. Timeout Routine. It returns to Rx. Interrupt Routine. Fig. 3.5.a3 shows the flow chart of the EOP Interrupt Routine.

iv) ACK. Timeout Routine - This subroutine is called from the EOP Interrupt Routine. It checks the Retransmission counter. If it does not exceed a threshold it initiates retransmission of the last frame else it indicates to the user that transmission attempt is given up. Fig. 3.5a.4 shows the flow chart of the Ack. timeout Interrupt.

v) Secondary Polling on EOP Timeout Routine - This routine is invoked on an EOP timeout Interrupt. It transmits a Polling frame to the secondary and initializes 8273 to generate an EOP character at the end of the Polling frame. It returns to the interrupted routine. Fig. 3.5a.5 shows its flow chart.

vi) Transmit Interrupt Routine - This routine is invoked after the completion of transmission of a frame. It stores the result byte of transmission and updates SOBT to indicate successful xmission or otherwise. It returns to the interrupted routine. Fig. 3.5a.6 shows the flow chart of the Transmit Interrupt Routine.

vii) Acknowledgement Transmit Routine - This routine is invoked from the Main Decision Routine when there is an outstanding acknowledgement to transmit. It transmits the Ack. frame and updates Ack. Data Structure. It returns to the Main Routine. Fig. 3.5a.7 shows the flow chart of the Acknowledgement Transmit Routine.

b) Message Processing Routine - These routines are involved in the actual processing of message as well as command frames after they are received. They are as follows :

i) Principal Message Processing Routine

ii) Command Processing Routine

iii) Routine for processing secondary to secondary transfer in the Primary

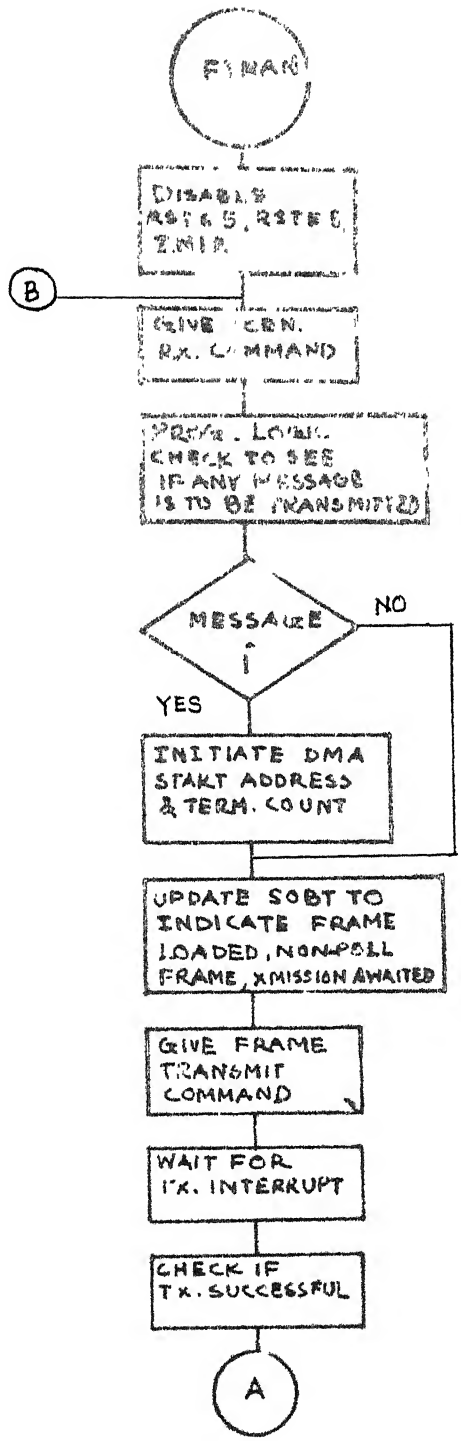


FIG: 3.5a.1 FRAME TRANSMIT ROUTINE: FTRAN

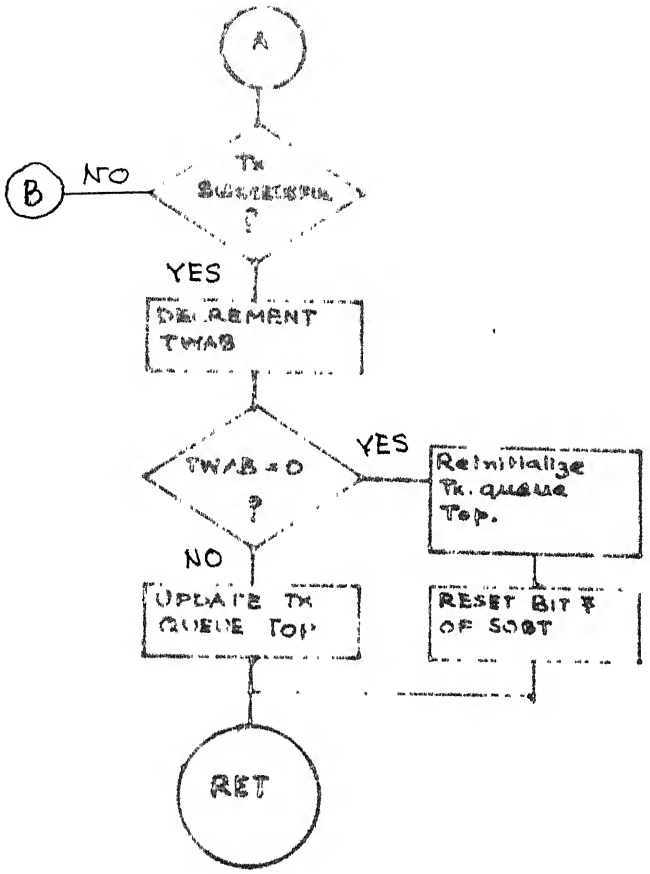


FIG : 3.5a.1 [CONTINUED]

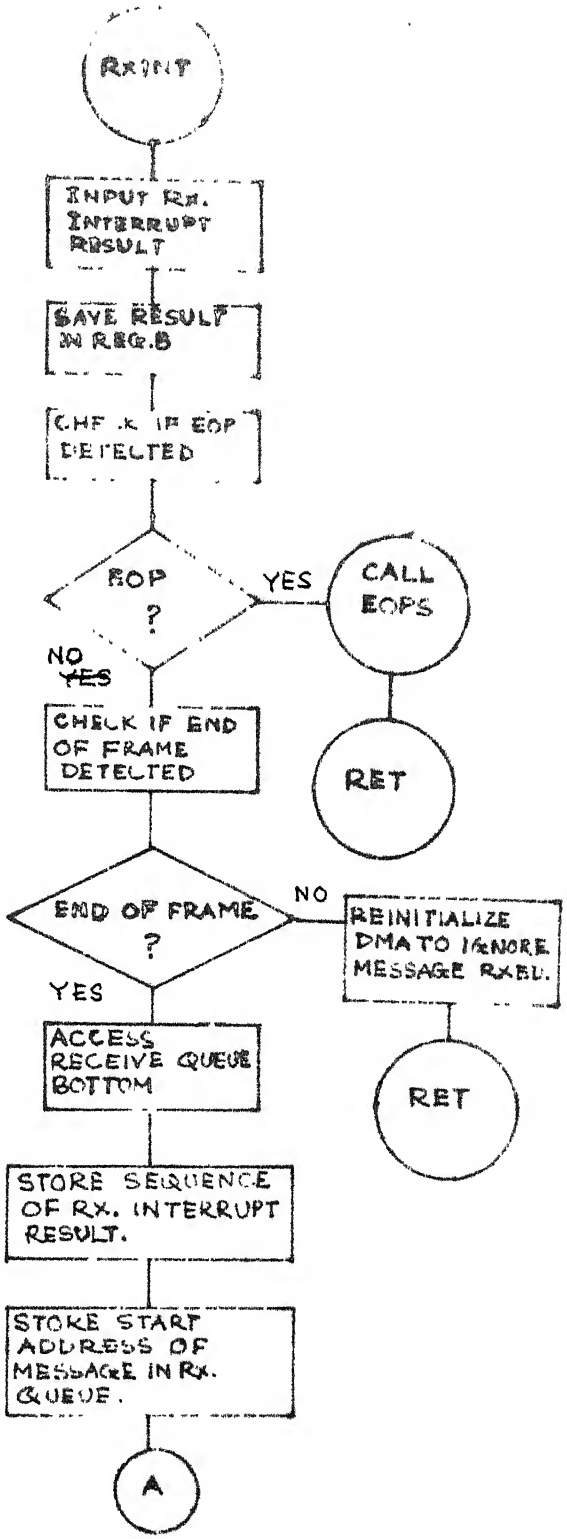


FIG : 3.5a.1 RECEIVE INTERRUPT ROUTINE : RxINT

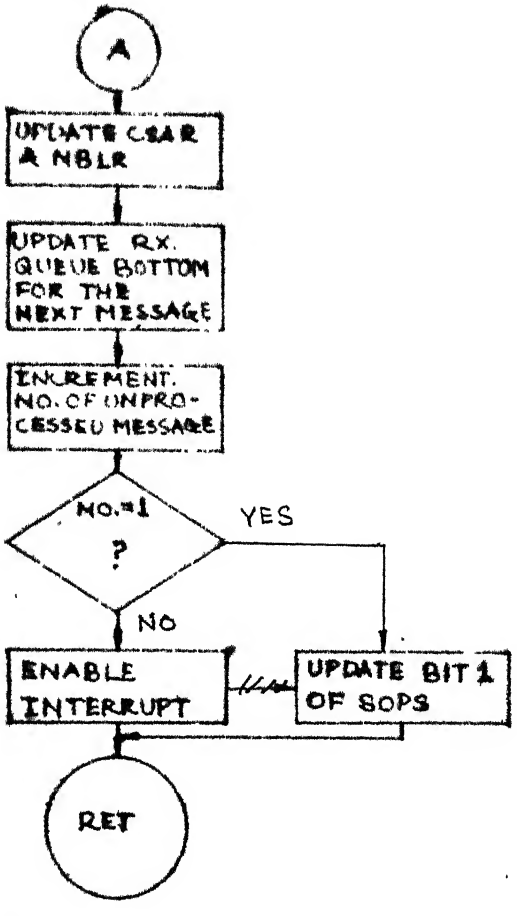


FIG: 3.5a.2 [CONTD.]

FIG 3.5a.3 EOP INTERRUPT ROUTINE : EOPS

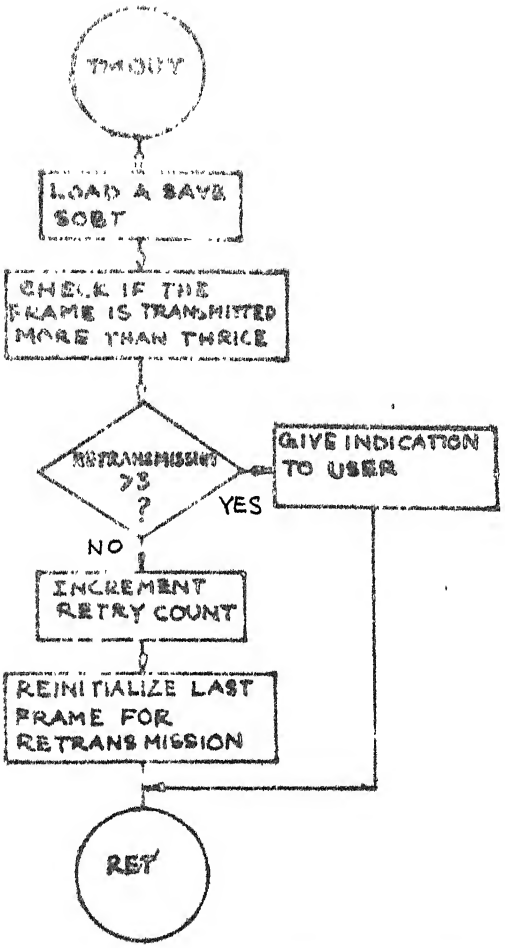


FIG: 3.5a.4 · ACK. TIMEOUT ROUTINE: TMOUT

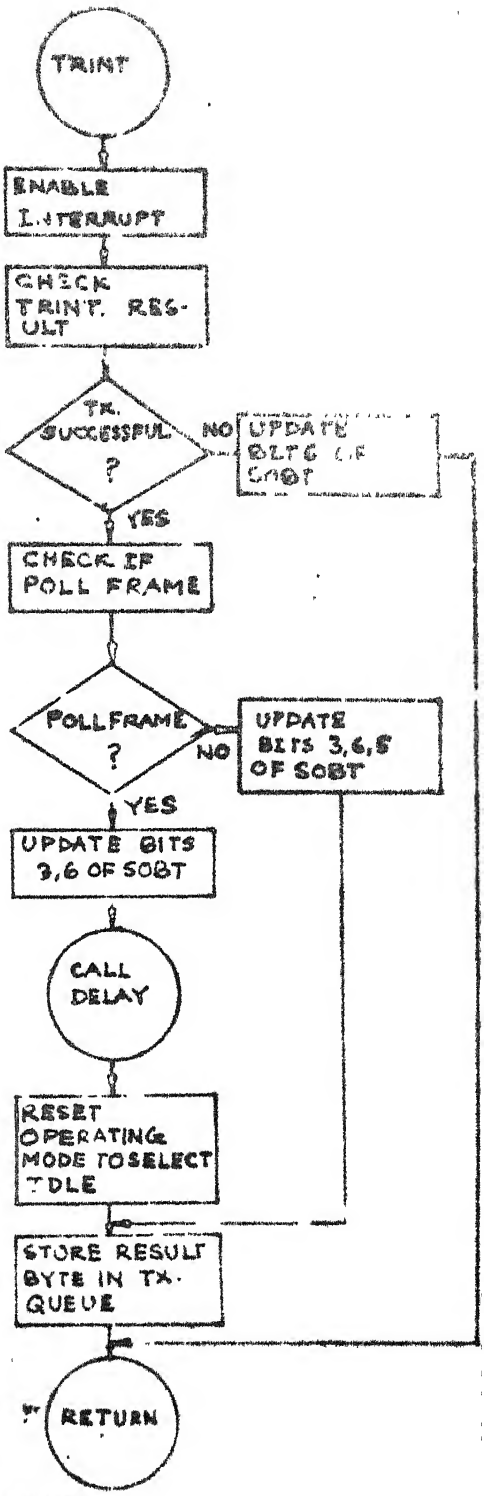


FIG: 3.5a.6 TRANSMIT INTERRUPT ROUTINE: TRINT.

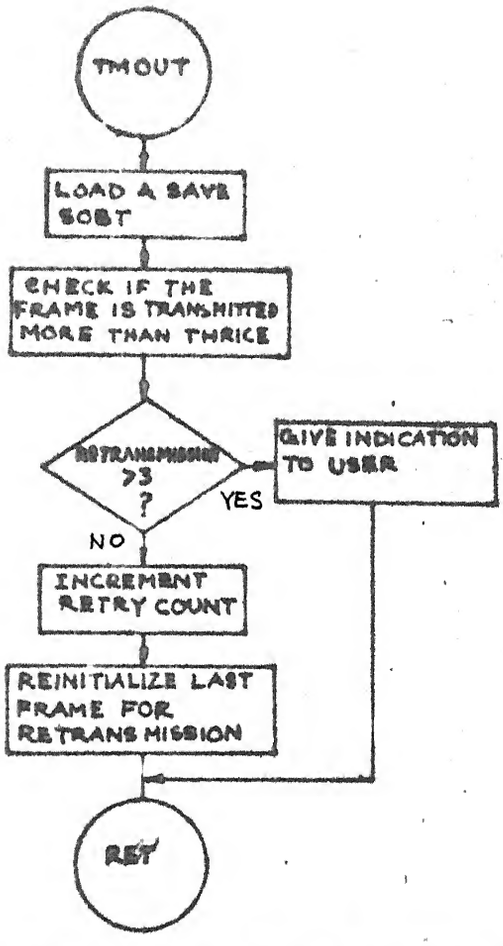


FIG: 3.5a.4 'ACK. TIMEOUT ROUTINE: TMOUT

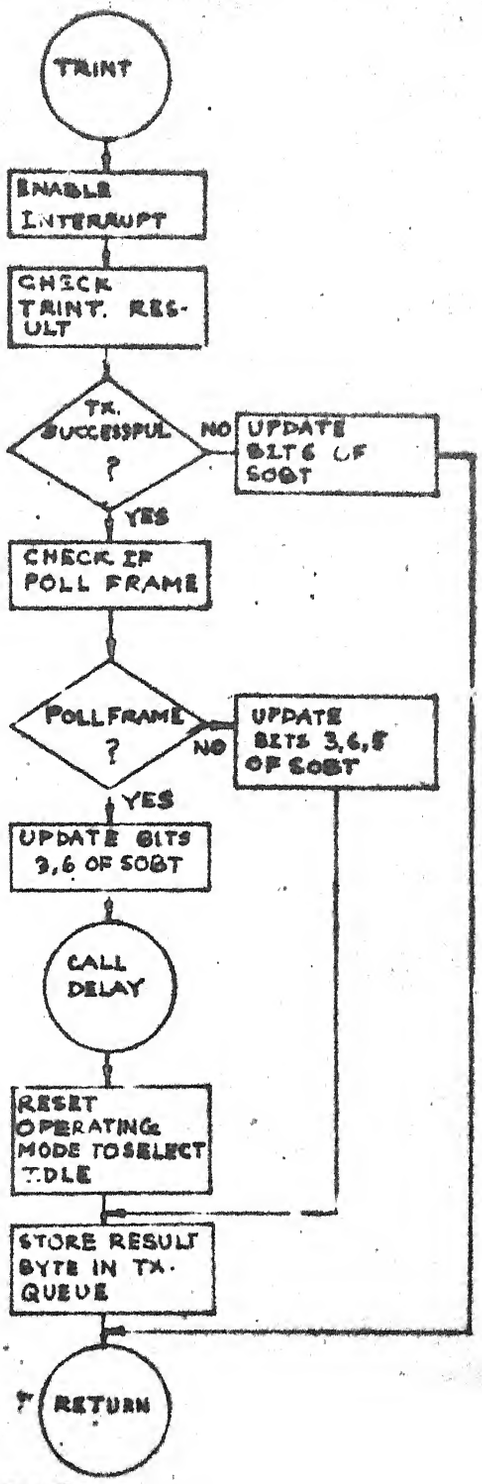


FIG: 3.5a.6 TRANSMIT INTERRUPT ROUTINE: TRINT.

iv) Routine for Updating Display Queue

v) Routine for Displaying Message to User

vi) Various Command Service Routines.

i) Principal Message Processing Routine - This routine is called from the Main routine whenever there is an unprocessed message in the Rx. queue. It processes messages as well as commands and its response is different depending upon the type of frame. For command processing it calls the command processing routine. Again for display of messages it calls the Display queue update routine. However, the common portion of the processing of all types of frame is carried out here. It returns to the Main routine. Fig. 3.5b.1 shows the flow chart of this routine.

ii) Command Processing Routine - This routine is called from the Pr. Message Processing Routine, if a command frame is received. It checks the control code and jumps to the appropriate command service routine (listed as vi) above). Fig. 3.5b.2 shows the flow chart of this routine.

iii) Routine for Processing Secondary to Secondary Transfer in Primary - This routine processes the secondary to secondary frames which are routed through the Primary. We should mention a point of note about the secondary to secondary frame processing. The primary changes the destination address of this frame substituting the original address with the implied destination address. It also updates a bit in the control byte to indicate

that the sender is primary. But the frame does not get retransmitted through the normal Tx. buffer update sequence. Instead, this frame is rerouted through as a Dummy Acknowledgement from the Primary to the destination secondary. This is done to avoid having to generate a separate Ack. frame for the received S.S. frame on the one hand and to avoid the necessity of the destination secondary station having to acknowledge the rerouted frame to the Primary, on the other. The dummy acknowledgement saves the additional overhead of intermediate acknowledgement on the SS frames. The flow chart of this routine is shown in Fig. 3.5b.3.

iv) Routine for Updating Display Queue - This routine is called from the Pr. Message Processing Routine. It uptakes the display queue with the start address of the current message. Fig. 3.5b.4 shows the flow chart of this routine. It returns to the calling routine.

v) Routine for Displaying Message to User - This routine is invoked from the main program, whenever there is an entry in the display queue. It checks whether there is a display lock or not. If the user is being served through keyboard interrupt, display lock will be set and this routine returns to the main program. Else it displays the message, updates display queue and returns. Fig. 3.5b.5 shows the flow chart.

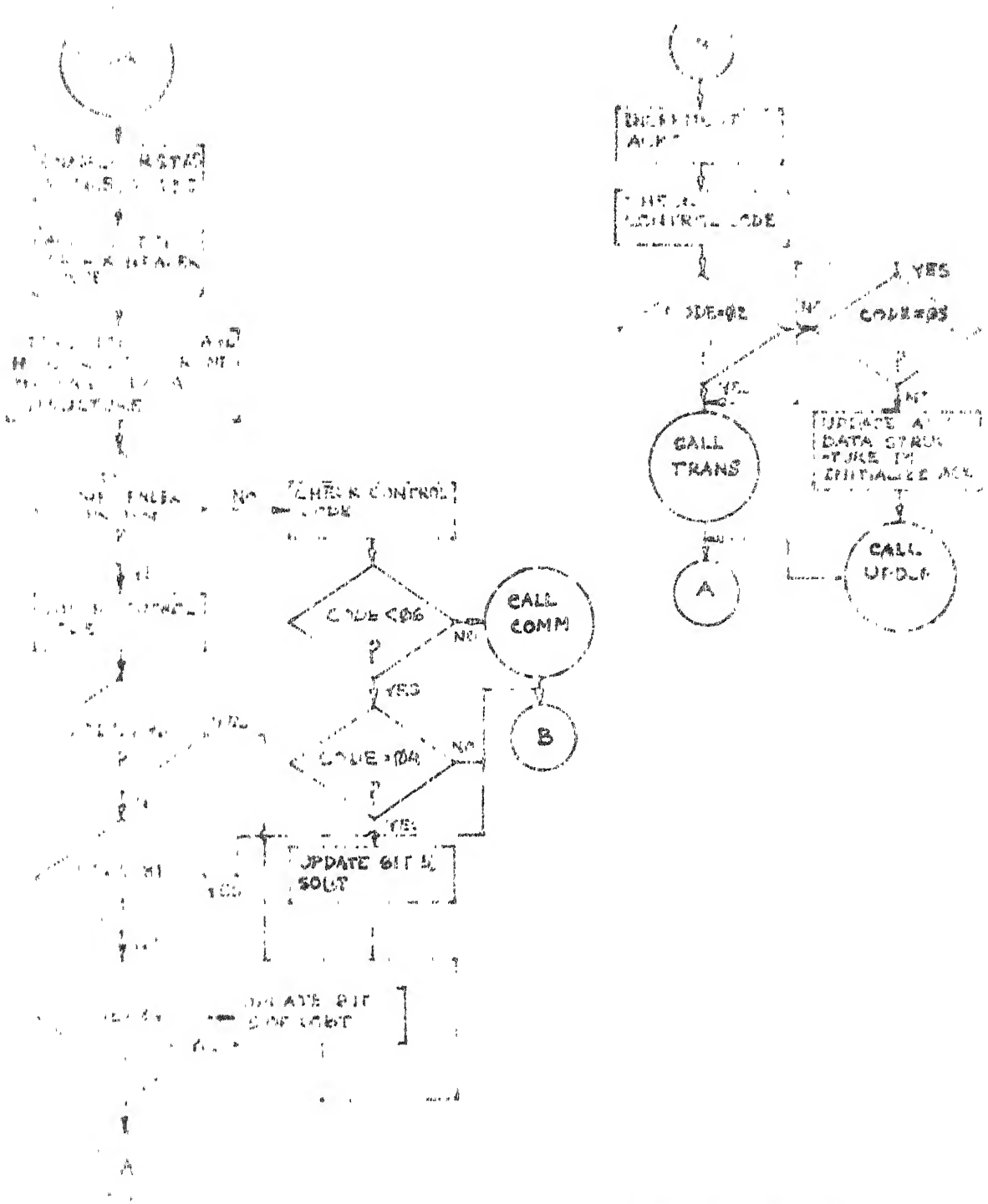
vi) Various Command Service Routines - The primary services the command frames sent by secondaries through these routines. They are invoked from the command processing routine. In our implementation we have designed only two secondary to Primary command frames viz. SPC and SRC. Fig. 3.5b.6 shows their flow charts.

c) User Side Routines - These routines cater directly or indirectly to the user. They are as follows :

i) Command Interrupt Routine
ii) Subroutine to decide the mode of TTY response
iii) Subroutine for TTY response in Mode 0
iv) Subroutine for TTY response in Mode 1
v) Subroutine for TTY response in Mode 2
vi) The various user command routines :

1) Store, 2) Display, 3) Transmit, 4) Restore, 5) Reset
6) Online Secondary, 7) Disconnect Secondary,
8) Status Poll, and 9) Quit.

i) Command Interrupt Routine - This routine is the RST 5.5 service routine. It is invoked by pressing of any key on the keyboard. It sets the display lock and calls the routine for selecting the mode of TTY interrupt. Fig. 3.5c.1 shows its flow chart.


YES
CALL
TRANS
A
CALL
COMM
B
YES
YES
A

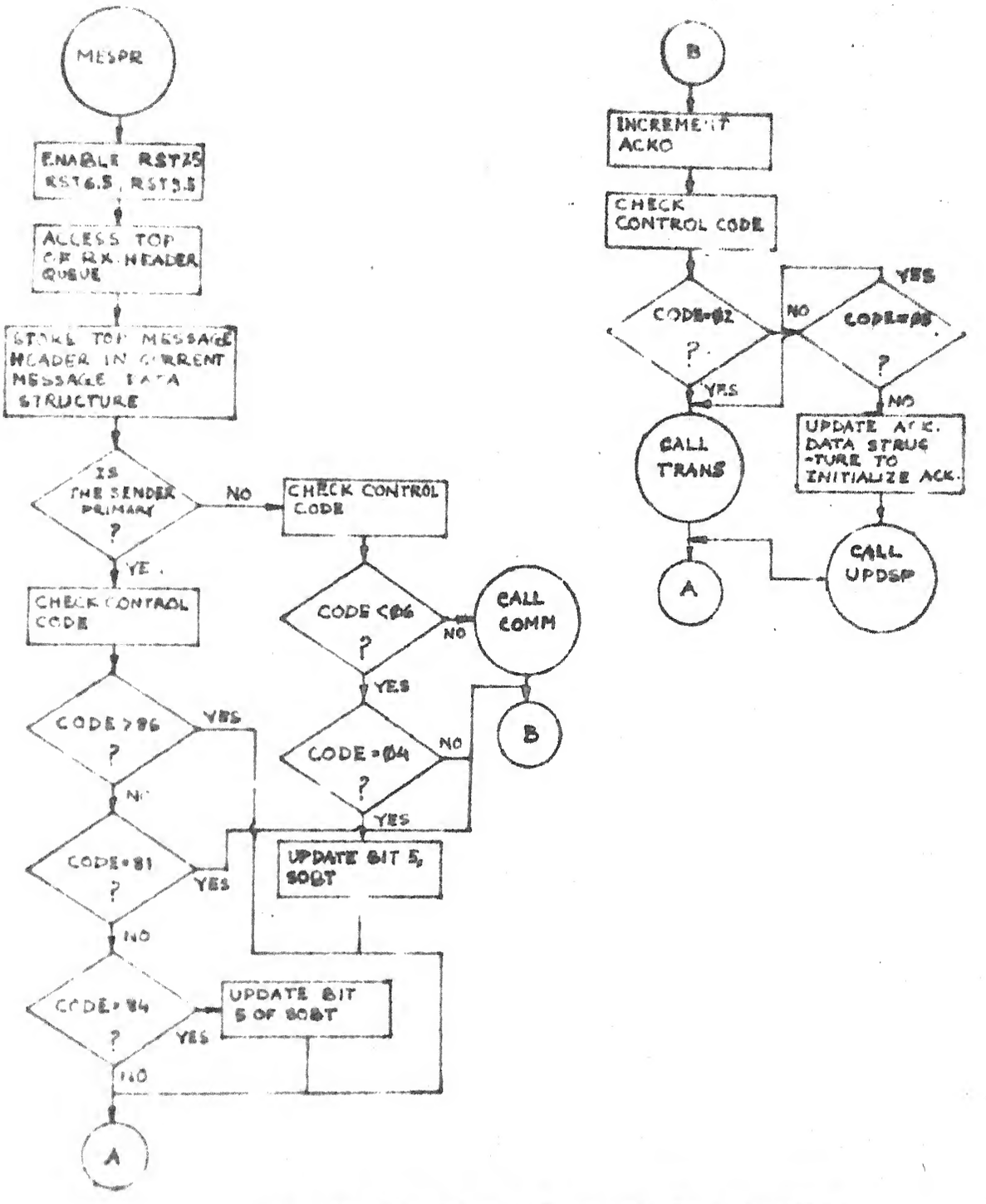


FIG: PRINCIPAL MESSAGE PROCESSING ROUTINE : MESPR
(3 SHT)

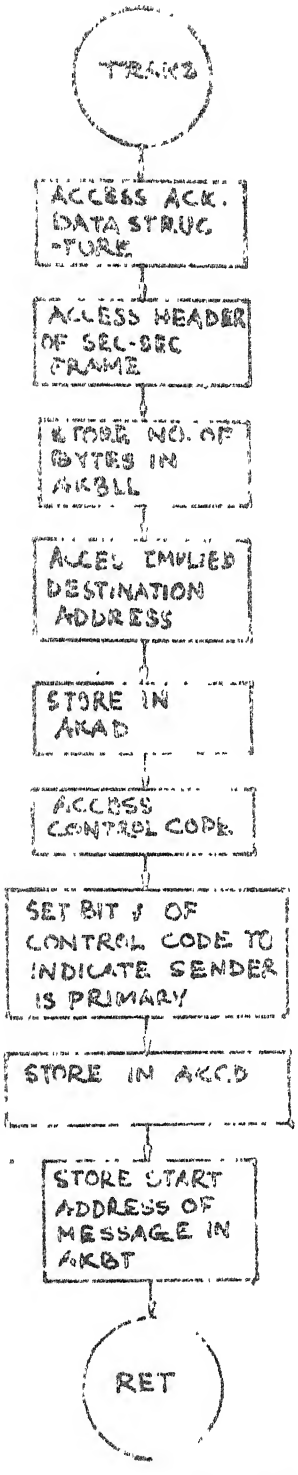


FIG: 5.10 SEC-SEC FRAME PROCESSING ROUTINE : TRAN2

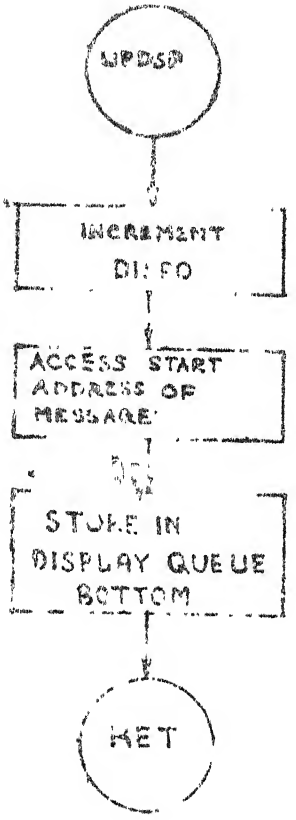


FIG: 3.56.4 ROUTINE FOR UPDATING DISPLAY QUEUE

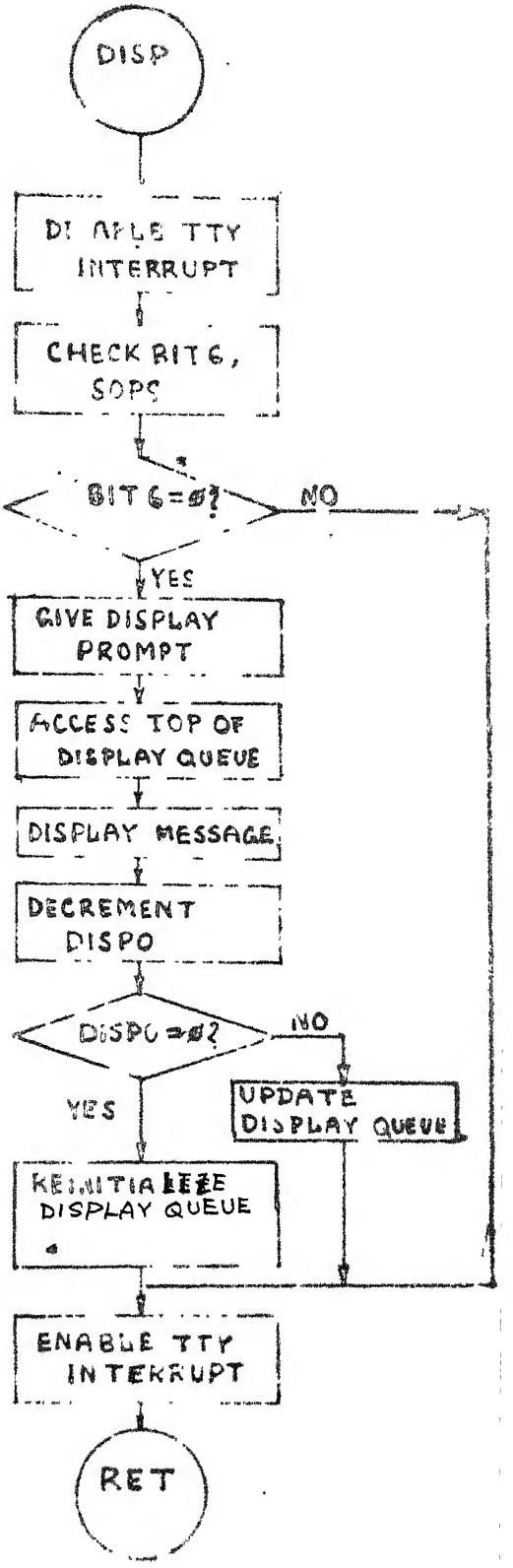
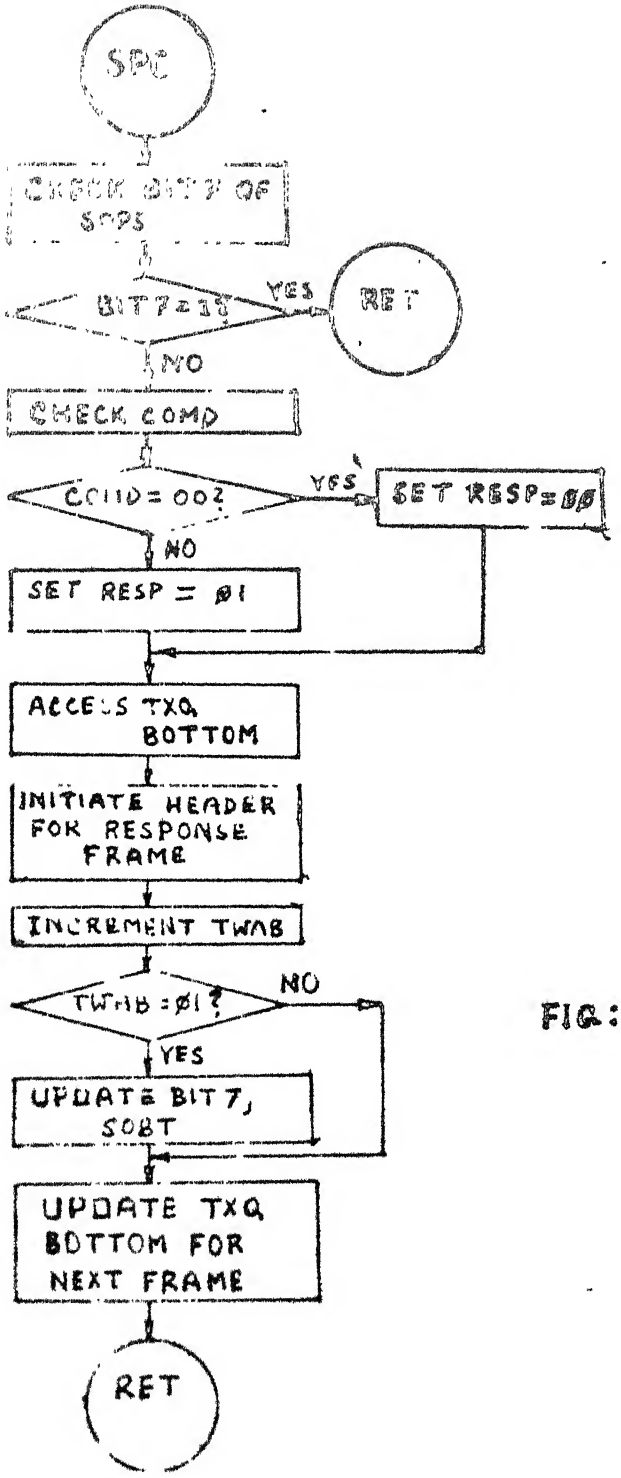


FIG: 3.56.5 ROUTINE FOR DISPLAYING MESSAGE TO USER



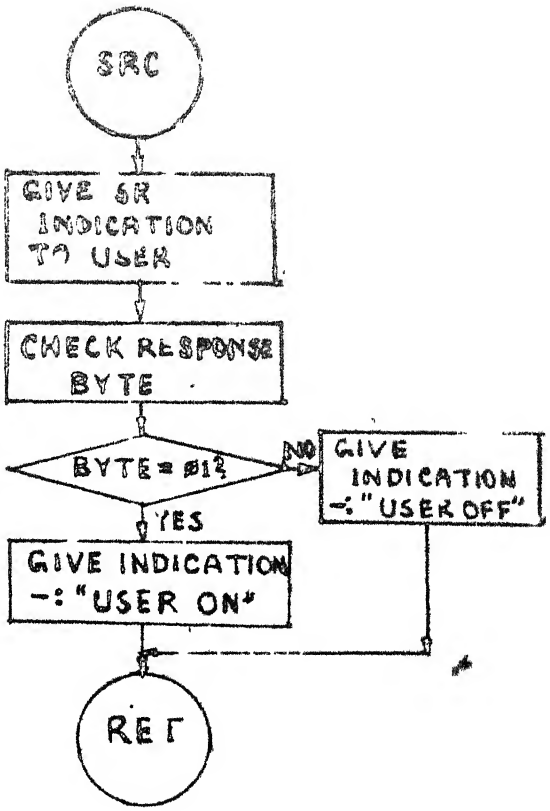


FIG: 3-5 b.6 COMMAND SERVICE ROUTINES SPC & GPC

ii) Subroutine to decide the mode of TTY interrupt - This subroutine is called from the Command Interrupt Routine. It decides whether the station is in Mode 0 (i.e. idle mode), Mode 1 (i.e., Command processing mode) or Mode 2 (i.e. Data storage mode). It returns to the calling routine. Fig.3.5c.2 shows the flow chart of this routine.

iii), iv) and v) Subroutines for TTY response in Mode 0, Mode 1 and Mode 2 respectively - The flow charts for these routines are given in Fig. 2.5c.3, Fig. 3.5c.4 and Fig. 3.5c.5 respectively.

vi)

1) Store - This routine checks whether the station is in Mode 2. If yes, then it issues the store prompt to the TTY and initializes store pointers. It returns to the TTY response routine iv). The flow chart is given in Fig. 3.5c.6(i).

2) Display - It displays the currently stored message in TX buffer. It returns to TTY response routine iv). The flow chart is given in Fig. 3.5c.6(ii).

3) Transmit - It creates the Transmit header of the frame stored currently in TX. buffer and initiates it for transmission. It returns to TTY response routine, iv) Flow chart is given in Fig. 3.5c.6(iii).

4) Restore - It restores the previously transmitted frame in the TX buffer. Flow chart is given in Fig. 3.5c.6(iv).

5) Reset - It checks the destination address in the command. If the address is that of Primary, it jumps to the Primary initialization routine after disabling interrupt. Else it sends the command frame corresponding to Reset to the destination secondary. Flow chart is given in Fig. 3.5c.6(v).

6) Online Secondary - The Primary sends a frame with the control code of online secondary to the destination address. The flow chart is given in Fig. 3.5c.6(vi). The flow chart of online secondary, disconnect secondary and status poll are almost identical except for the difference in control code. Thus in fact, all three of them jump to the same routine segment (SC) after incorporating the proper control code.

7) Disconnect Secondary - The Primary initiates a frame having control code of Disconnect Secondary to transmit to the destination address. The flow chart of this routine also is shown in Fig. 3.5c.6(vi).

8) Status Poll - Similar to the above two cases. It sends a control code of status poll to the destination address specified in the command. The flow chart of this also is shown in Fig. 3.5c.6(vi).

9) Quit - This routine simply takes the station out of command service mode and places it in the idle mode. The user will have to login again to reenter command service mode. Fig. 3.5c.6(vii) shows the flow chart.

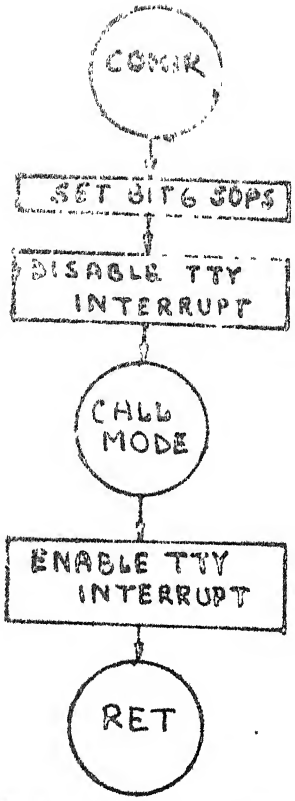


FIG: 3·5C.1 COMMAND INTERRUPT ROUTINE: COMIR

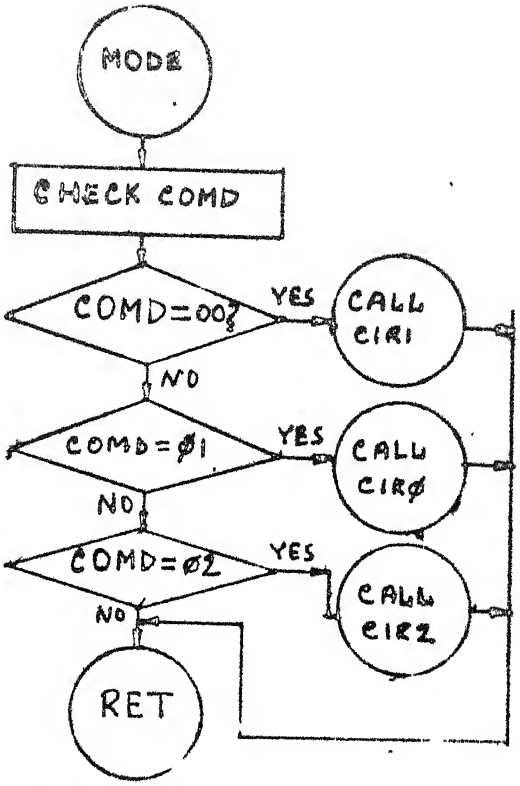


FIG: 3·5C.2 SUBROUTINE TO DECIDE MODE OF TTY INTERRUPT RTN.: MODE

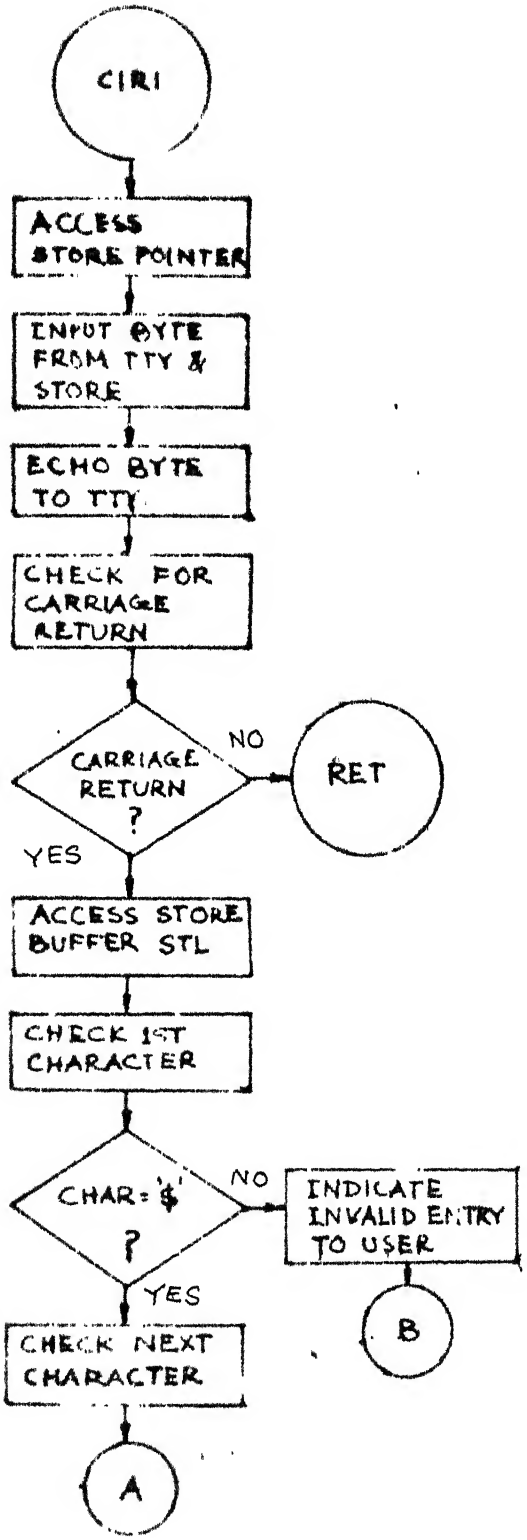


Fig. 3.5C.3 TTY INTERRUPT ROUTINE IN MODE0 : CIR1.

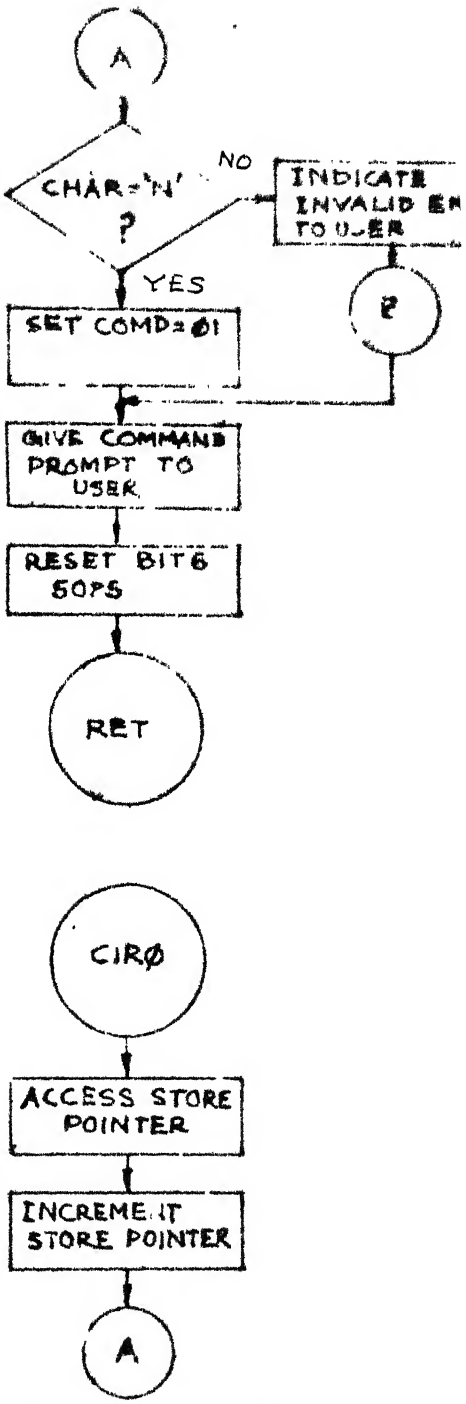


Fig: 3.5C.4 TTY INTERRUPT ROUTINE IN MODE1 : CIR0

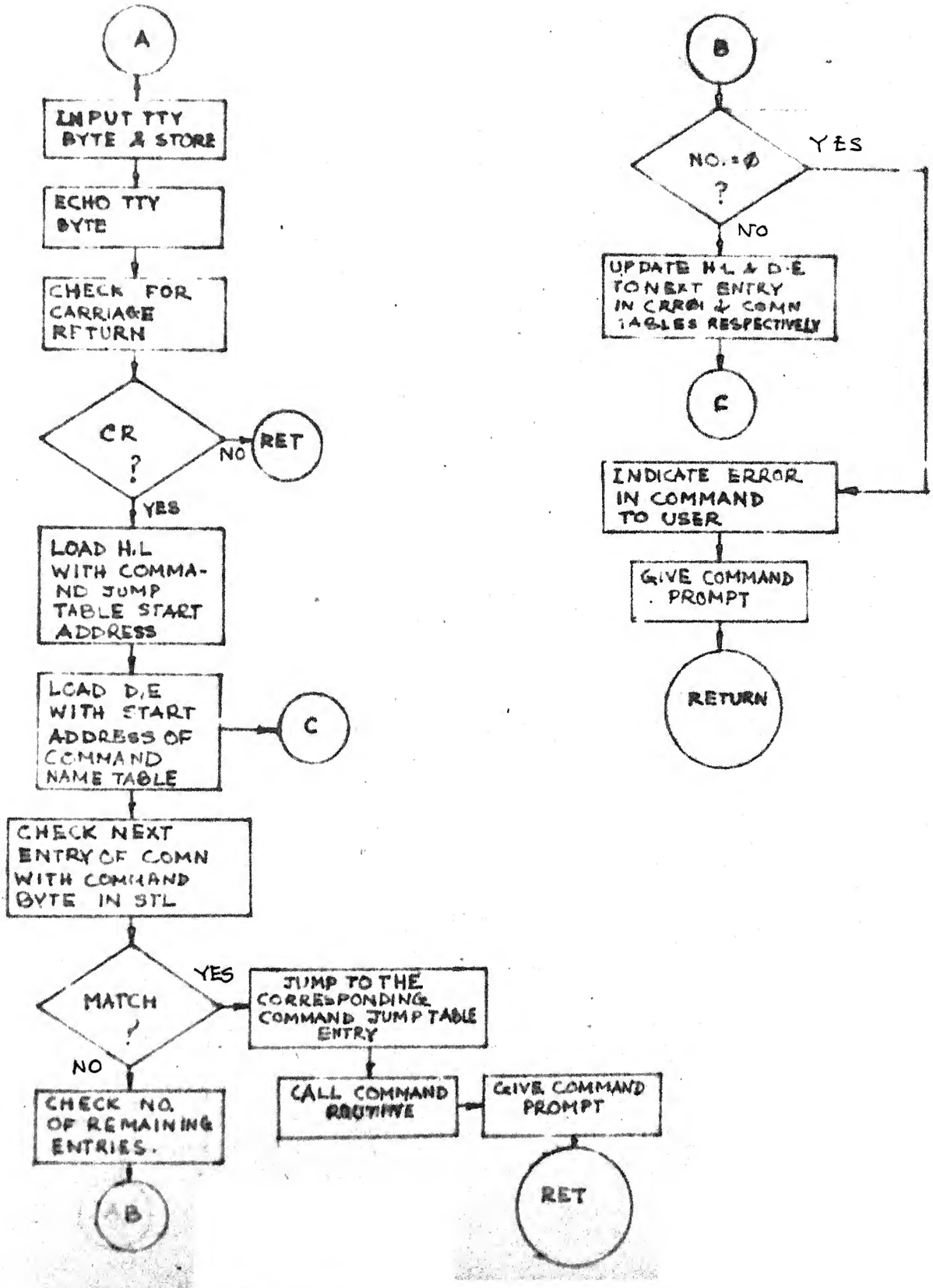


Fig: 3.9c4 [CONTD]

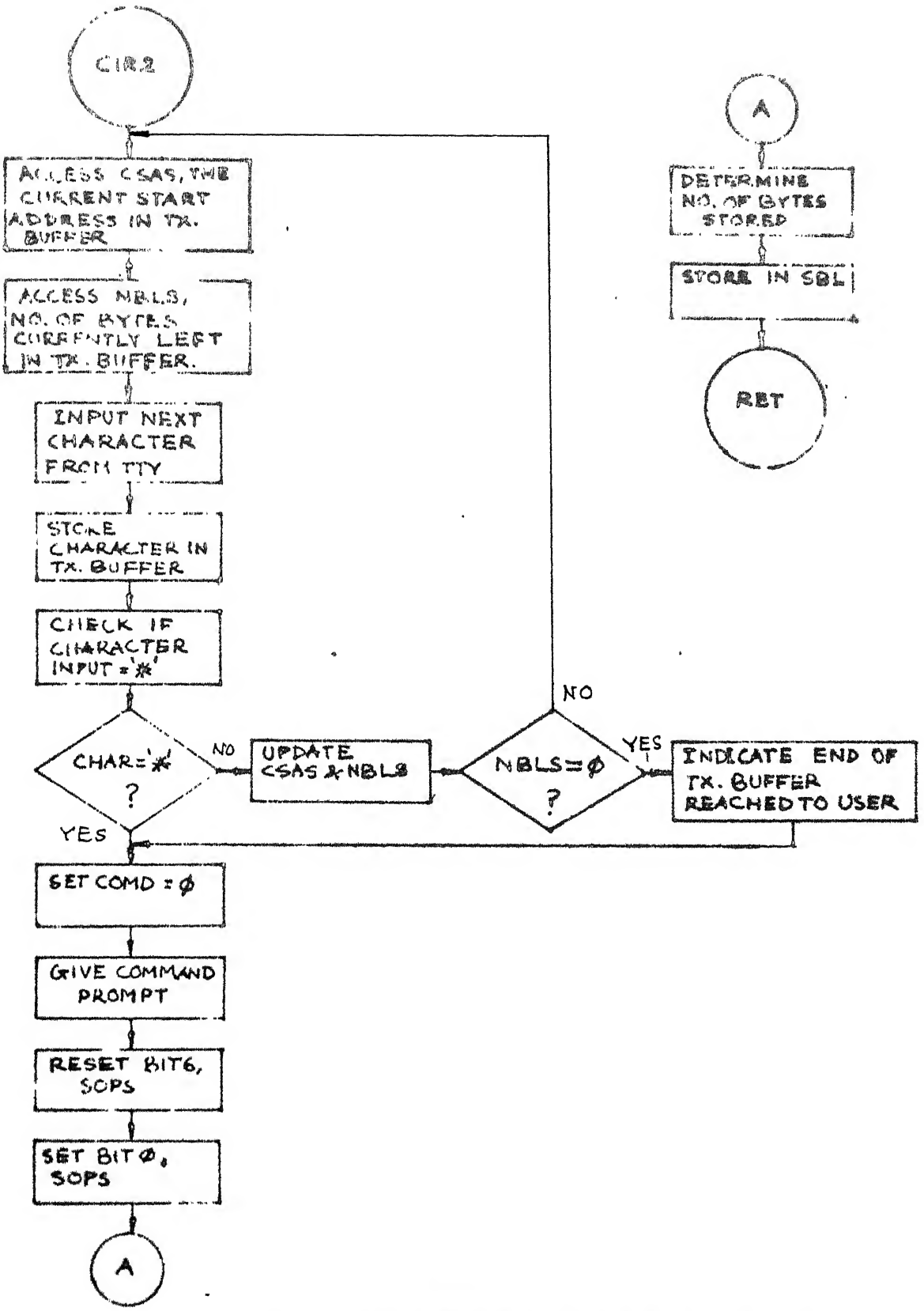


Fig: 3.6C.5 TTY INTERRUPT ROUTINE IN MODE 2: CIR2

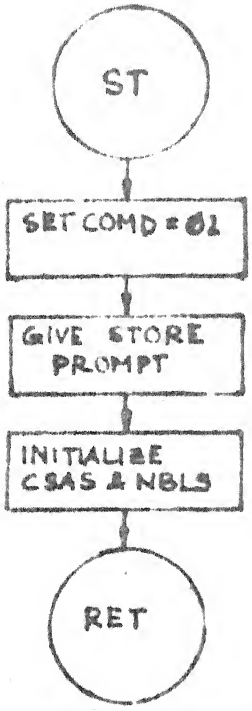


FIG: 3.5C.6(i) STORE ROUTINE: ST

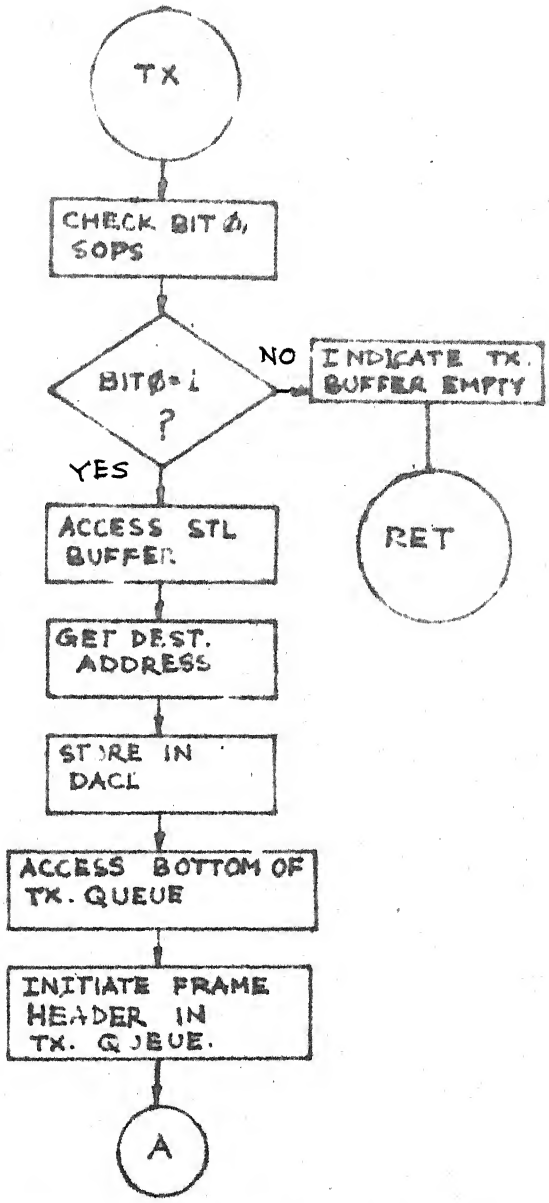


FIG: 3.5C.6(ii) TRANSMIT ROUTINE : TX

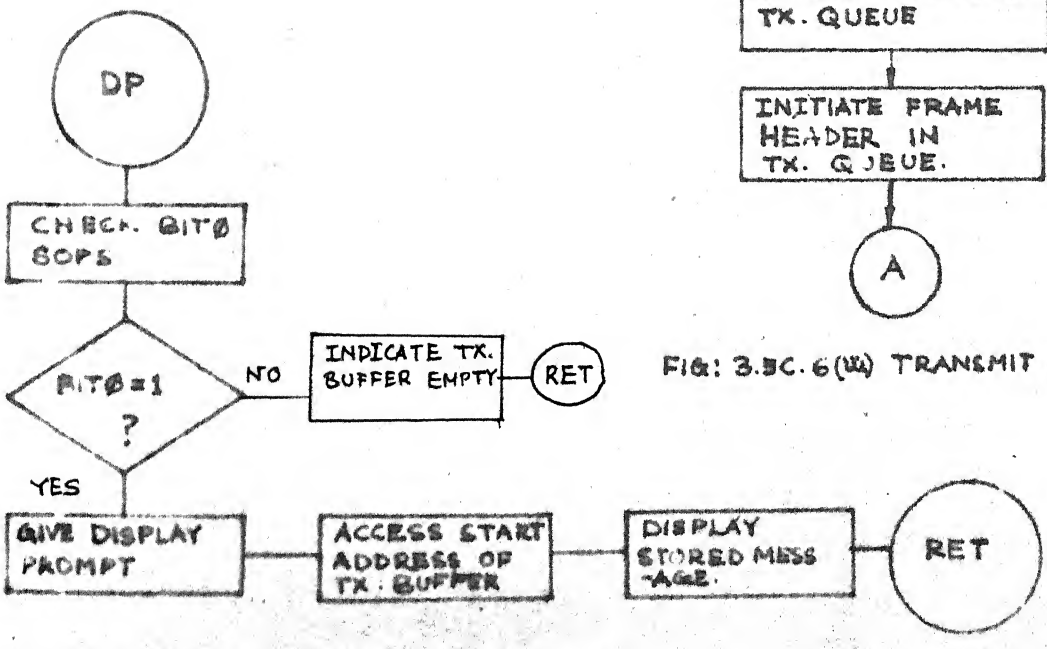


FIG. 3.5C.6(iii) DISPLAY ROUTINE : DP

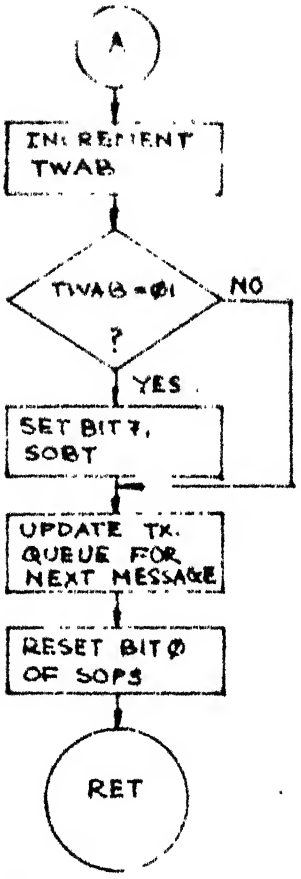


FIG: 3.5.6(iii) [CONTD.]

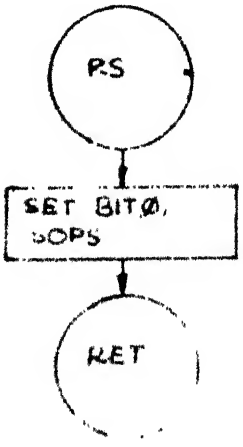


FIG: 3.5.6(iv) RESTORE ROUTINE: RS

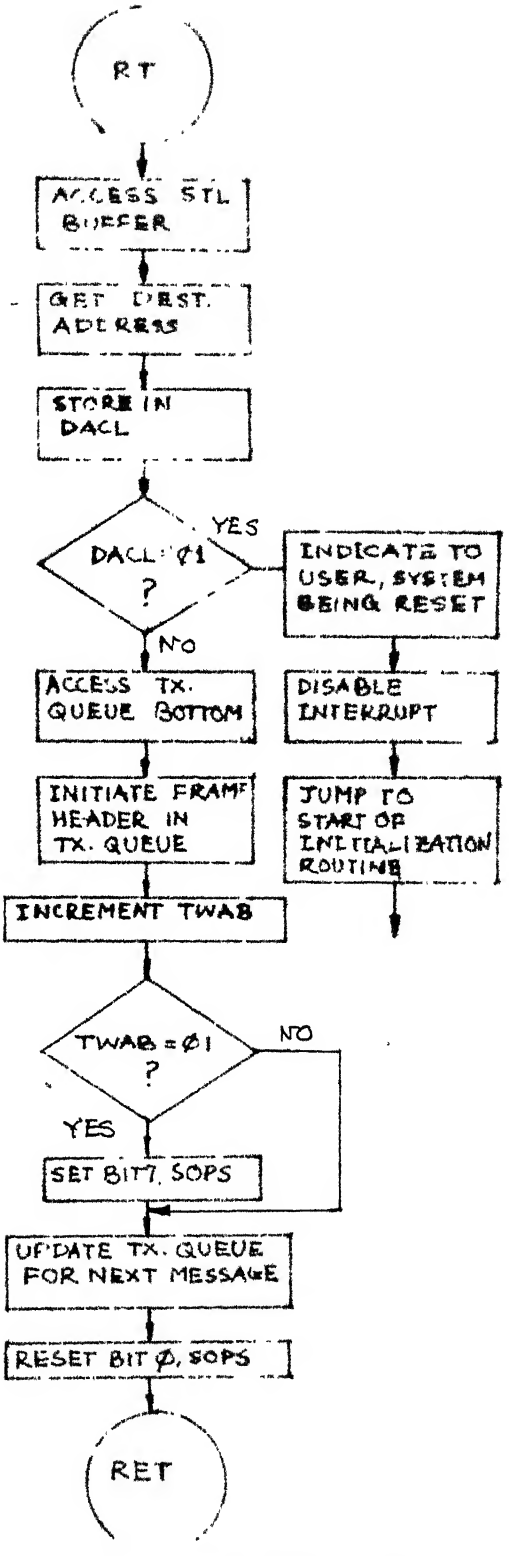


FIG. 3.5.6(v) RESET ROUTINE: RT

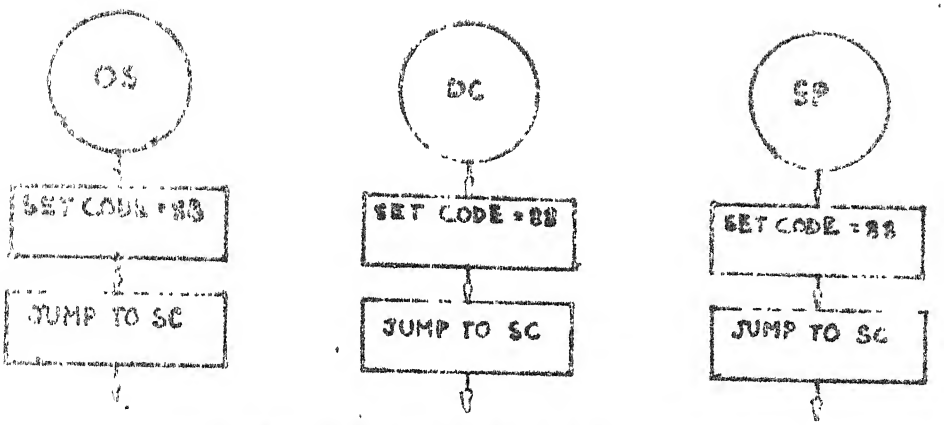


FIG: 3.5C.6(iv) { ONLINE SECONDARY : OS
DISCONNECT SECONDARY : DC
STATUS POLL : SP

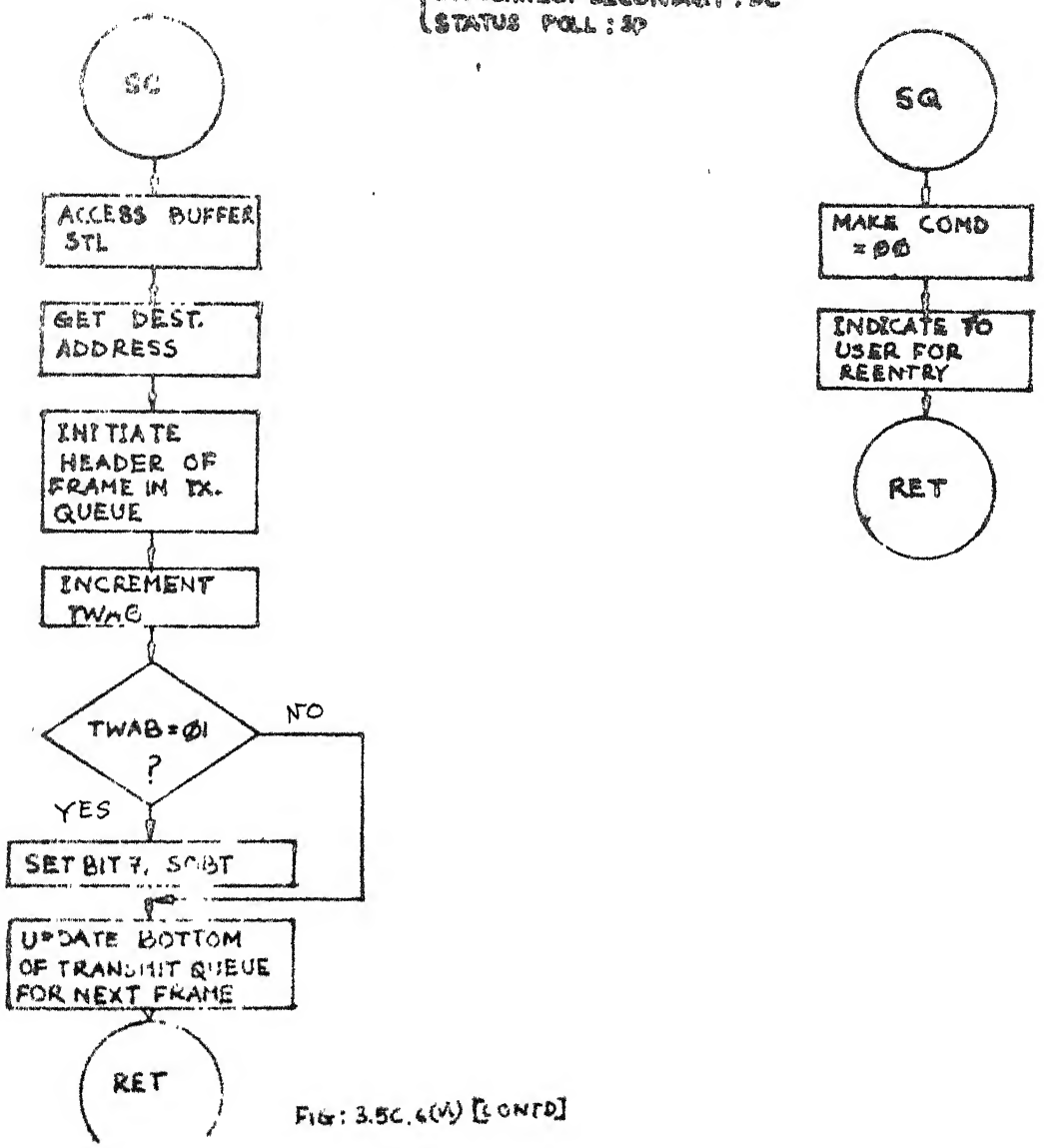


FIG: 3.5C.6(v) [CONTD]

d) Loop Primary Main Program - This is the main decision routine of the Primary. It controls the execution path of the Primary software. Fig. 3.5d.1 shows the flow chart of this routine.

3.6 DEVELOPMENT OF SECONDARY SOFTWARE

The secondary has a simpler but somewhat different program logic. The secondary is initialized into receive mode. It accepts whatever frame is sent to it having a frame address identical to its own address. At the same time it repeats it in one bit delay mode to down loop secondaries. However, if a Primary sends a Polling frame to the secondary then the secondary terminals receive mode, if it has any frame to transmit and captures the loop subsequently on getting a EOP character. After transmitting the frame it goes back into one bit delay RX mode. The Acknowledgement scheme is exactly identical to that of the Primary except for the fact that the secondary times out on the number of Polling frames received rather than on number of EOP received. This is because in secondary EOP interrupts are disabled.

On the user side, the secondary's function is exactly similar to that of the Primary and its processing of keyboard interrupt and response to user commands are all identical. However, its command set is smaller than that of the primary. Commands like Online Secondary, Disconnect, Reset etc. having supervisory nature are unique to the Primary only. This is

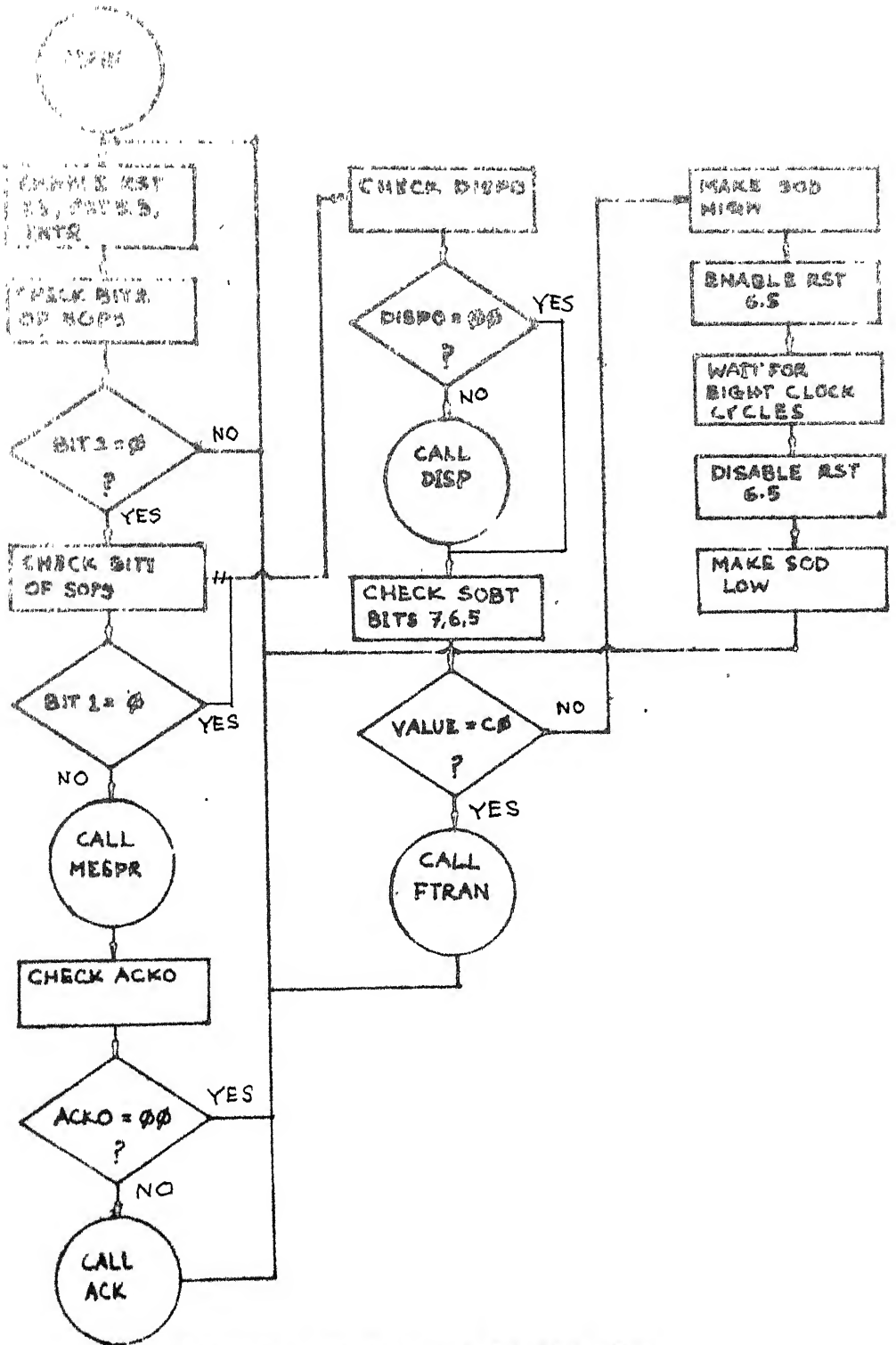


FIG: 3.5d.1 MAIN PRIMARY ROUTINE : MAIN

consistent with the master-slave organization of the network.

For describing the various routines of the secondary software we will adopt the following scheme. Since a majority of secondary routines follow those of the primary albeit with some superficial differences, we will just present the flow-charts of these routines without adding further explanatory notes. However, those few routines that are functionally specific to the secondary will be explained in detail along-with the flow charts.

Following is the list of main secondary routines :

a) Link side Routines :

i) Frame Transmit Routine

ii) RX. Interrupt Routine

iii) TX. Interrupt Routine

iv) Acknowledgement transmit Routine

v) Acknowledgement timeout Routines

vi) Dummy frame transmit Routine.

Routines i) to iv) are similar to those corresponding routines in Primary. The flow charts for these routines are given in Fig. 3.6a.1 to Fig. 3.6a.4. Routines v) is exactly identical with the corresponding routine in primary. Refer to Fig. 3.5a. 4 for its flow-chart.

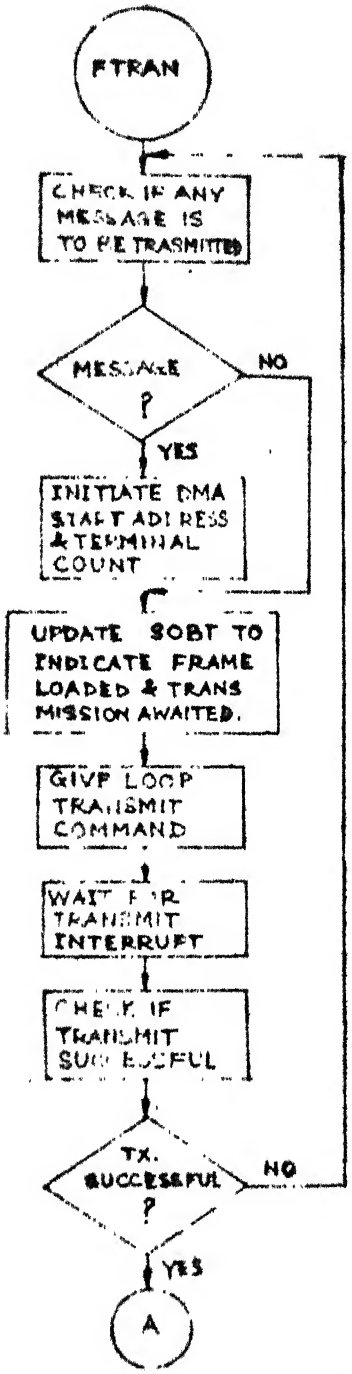


FIG: 3.6a.1 FRAME TRANSMIT ROUTINE: FTRAN

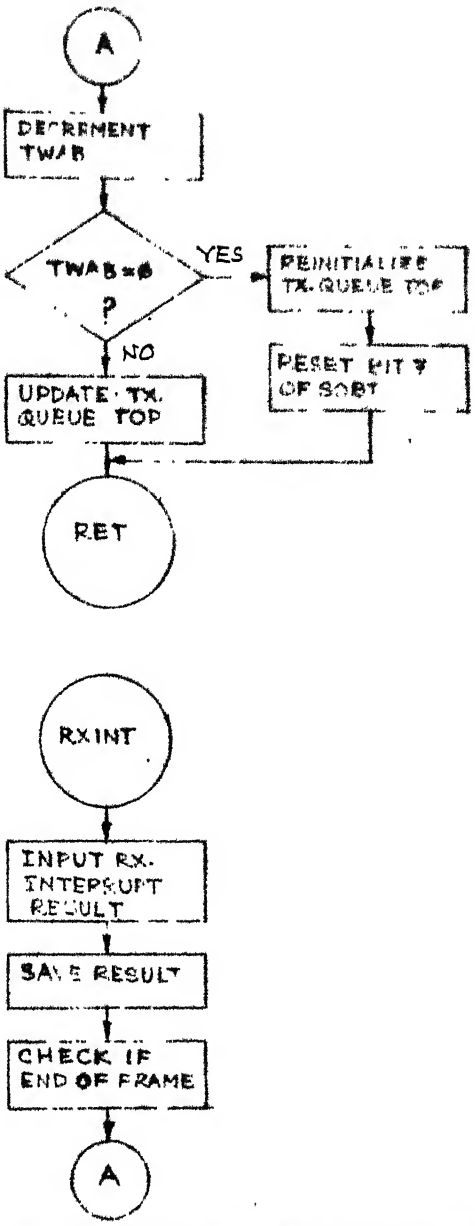


FIG: 3.6a.2 RX. INTERRUPT ROUTINE: RXINT

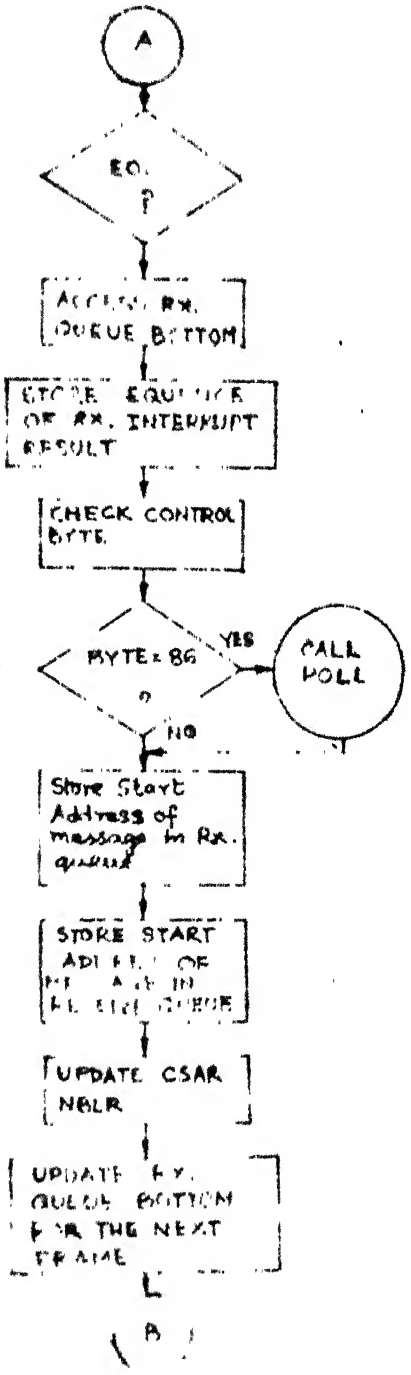
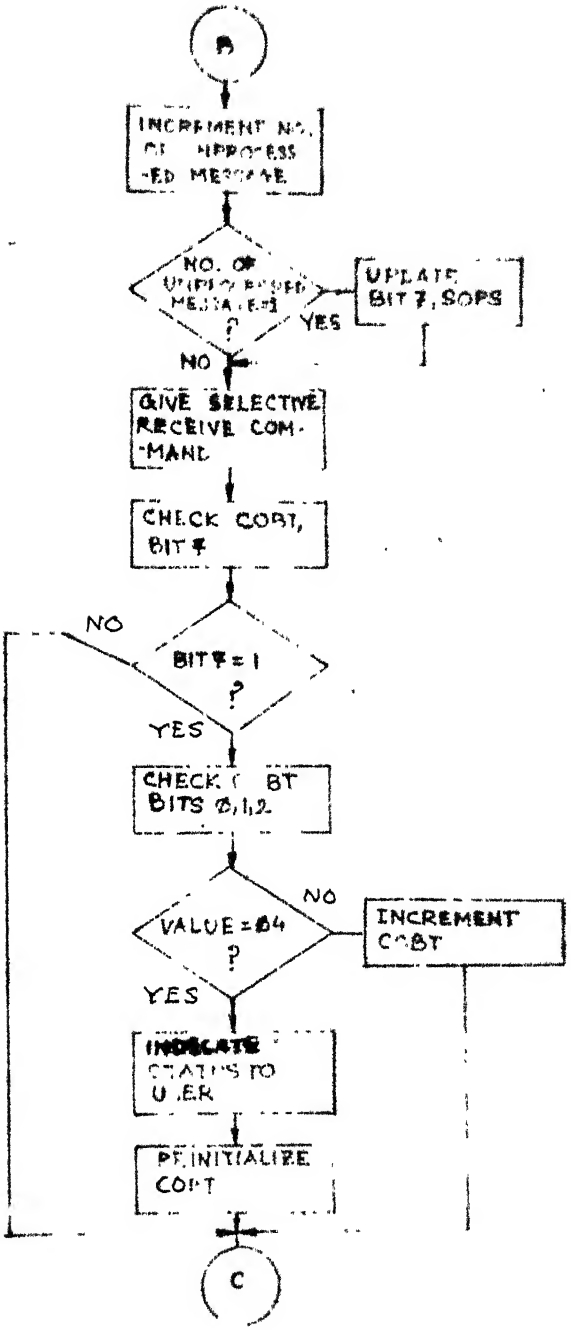

A
EQ.
?
ACCESS RX. QUEUE BOTTOM
STORE EQUIVALENCE OF RX. INTERRUPT RESULT
CHECK CONTROL BYTE
BYTE = 86
?
YES
CALL POLL
NO
Store Start Address of message in Rx. queue
STORE START ADDRESS OF
UPDATE CSAR NBLR
UPDATE RX. QUEUE POSITION FOR THE NEXT FRAME
B

FIG: 3.2 [CONTD]


B
INCREMENT NO. OF PROCESSED MESSAGE
NO. OF UNPROCESSED MESSAGE = 1
?
YES
UPDATE BIT 7, SOPS
NO
GIVE SELECTIVE RECEIVE COMMAND
CHECK COBT, BIT 7
BIT 7 = 1
?
NO
YES
CHECK COBT BITS 0,1,2
VALUE = 04
?
NO
INCREMENT COBT
YES
INDICATE STATUS TO USER
RE-INITIALIZE COBT
C

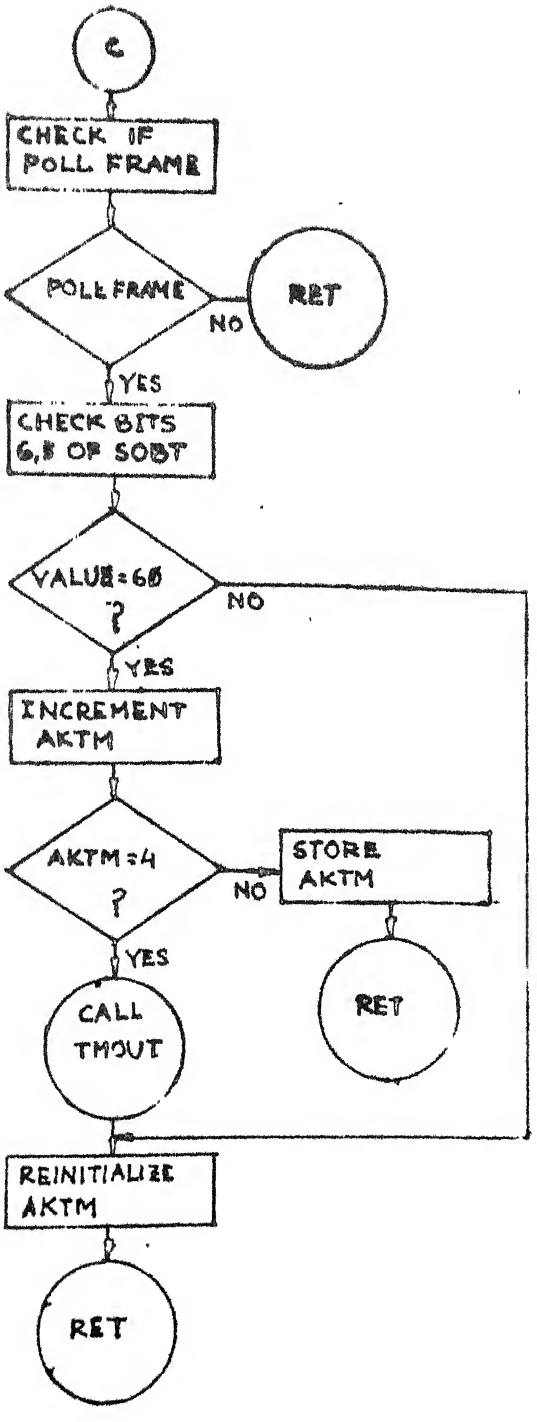


FIG: 3.6a.2 [CONTD.]

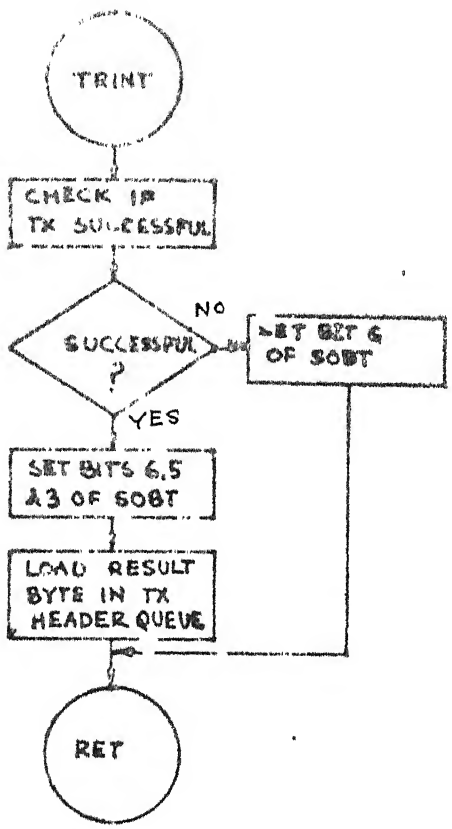


FIG: 3.6a.3 TRANSMIT INTERRUPT ROUTINE: TRINT

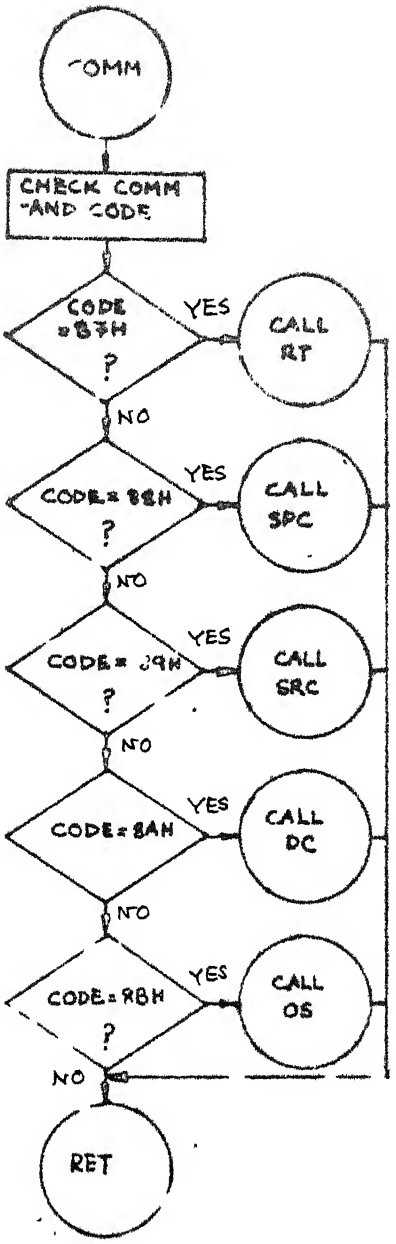


FIG: 3.6b.2 COMMAND PROCESSING ROUTINE: COMM

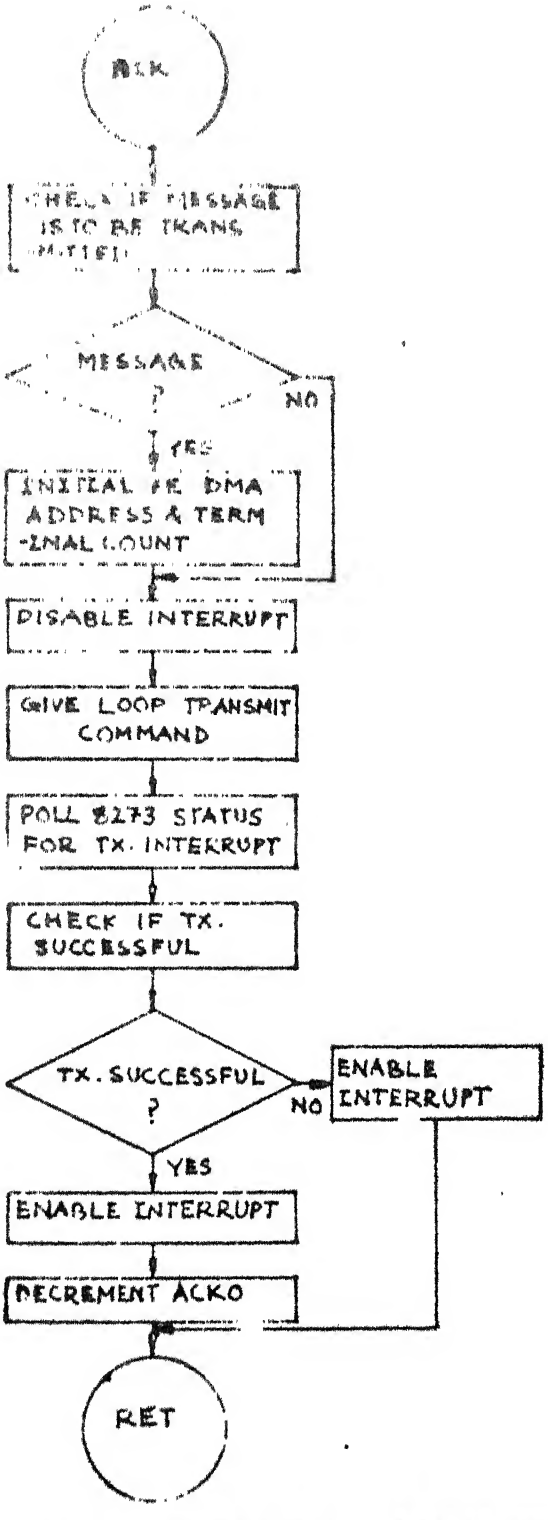


FIG: 3.6a.5 ACK. TRANSMIT ROUTINE: ACK.

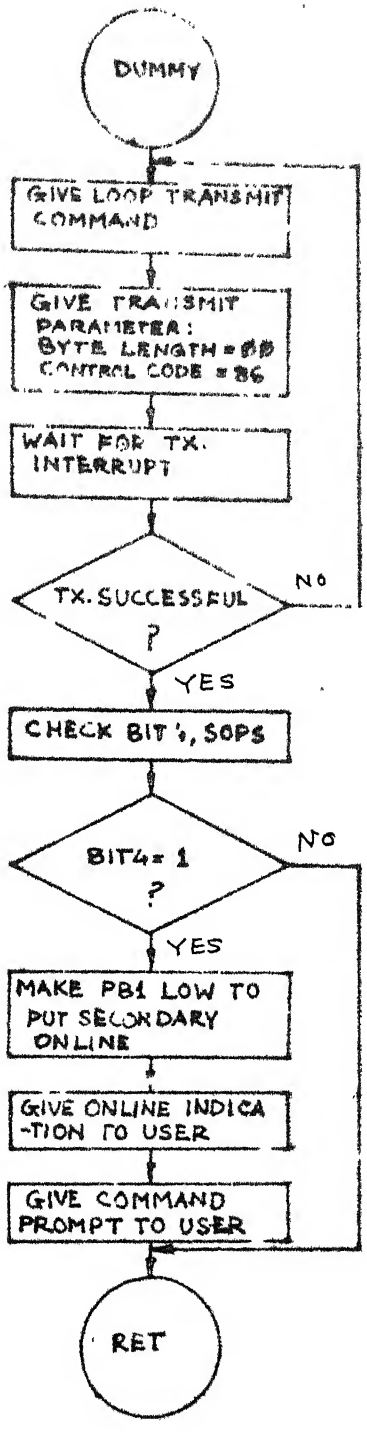


FIG: 3.6a.6 DUMMY FRAME TRANSMIT ROUTINE: DUMMY

Routine vi) is the Dummy Frame Transmit Routine - In the secondary, the sequence of event that happens after an EOP character is received is conditioned by the past history of the program execution. If the secondary was issued a selective loop Rx. Command prior to the arrival of the EOP character, then it is taken out of one bit delay mode. If it has a frame to transmit, then its transmission of the frame automatically brings the secondary back to one bit delay repeat mode. However, if no transmission is pending then we have to deliberately set operating mode of the secondary 8273 to bring it back to one bit delay mode or else the secondary keeps on transmitting flags to down loop stations. However, in trying to implement this scheme we found that the set operating mode command does not get accepted by the 8273 device at this point of operation. In order to get round this difficulty we decided to implement a scheme where the secondary would always have a frame to transmit in response to the polling frame. If it does not have a genuine frame to transmit, we would insert a dummy frame so that our objective of getting the 8273 device back in one bit delay mode is met. The dummy frame is essentially a frame of blank content with a control code which is rejected by the primary without processing. We used the control code of the polling frame itself to be used in the dummy frame. This ensures that we do not have to waste a control code to implement this scheme. The dummy frame gets rejected by the Primary in the same way as

the Polling frame does. Note, that the dummy frame has to be transmitted independently of the Txmit queue in order to bypass the acknowledgement scheme. This is precisely what the routine does.

Fig. 3.6a.5 shows the flow chart of this routine.

b) Message Processing Routines :

i) Principal Message Processing Routine

ii) Command Processing Routine

iii) Update Display queue Routine

iv) Routine to Display Message to User

v) Routine for processing EOP sequence

vi) Various command service routine.

Routines i) and ii) are similar to the corresponding routines of the primary. The flow chart of these two routines are given in Figs. 3.6b.1 and 3.6b.2.

Routines iv) and iv) are exactly identical to the corresponding Primary Routines. Hence, for their flow charts refer to Figs. 3.5b.4 and 3.5b.5.

v) Routine for processing EOP Sequence - This is a routine which decides the sequence of operation to be performed following the reception of Polling frame and upto the reception of the EOP character. The three possible ways in which secondary can respond to the polling frame are 1) by transmitting an ACK frame, 2) by transmitting a message/command frame and

3) by transmitting a dummy frame. Fig. 3.6b.3 shows the flow chart of this routine.

vi) Various command service Routines - The command service routines in the secondary are more numerous. There are in all five command service routines incorporated presently in the secondary and they correspond to the various commands designed in the primary. The command service routines are :

a) RT b) DC c) OS d) SPC and e) SRC

The last two mentioned are exactly identical to the corresponding Primary Routines. For their flow chart refer to Fig. 3.5b.6.

Routine RT resets the secondary on receiving a Command frame corresponding to reset command from the primary. Fig. 3.6b.4 shows its flow chart.

Routine DC disconnects the secondary from the loop and gives indication to the user about the same on receiving a Disconnect Command frame from the Primary. Fig. 3.6b.5 shows the flow chart of this routine.

Routine OS makes the secondary online to the loop and gives indication to the user on receiving an online command frame from the primary. Fig. 3.6b.6 shows the flow chart of this routine.

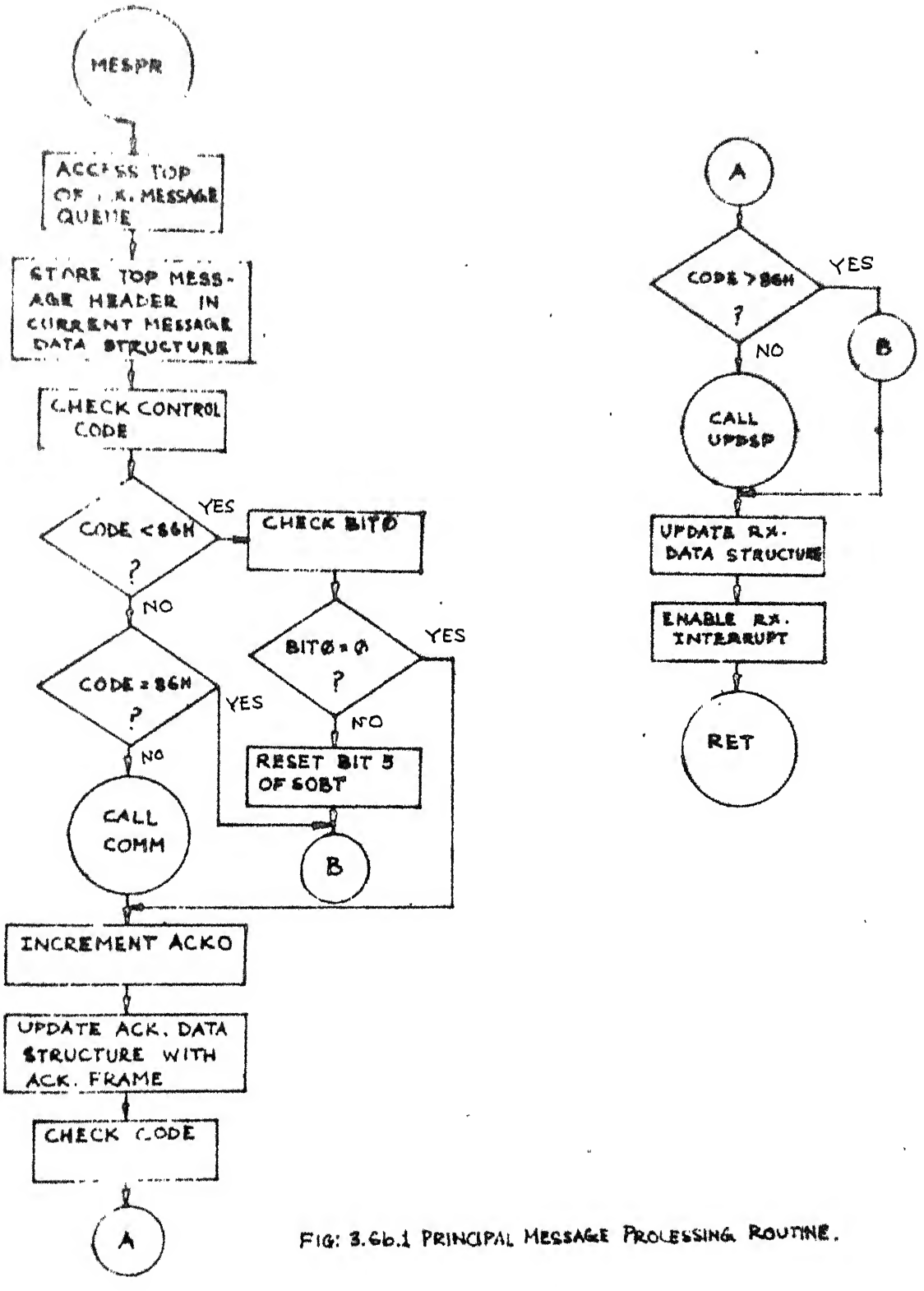


FIG: 3.6b.1 PRINCIPAL MESSAGE PROCESSING ROUTINE.

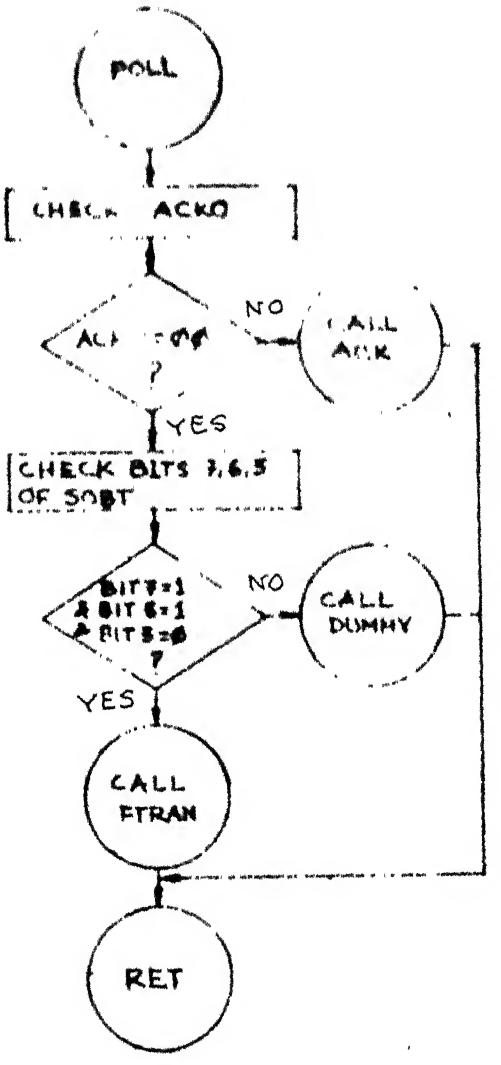


FIG: 3.66.3 SUBROUTINE TO PROCESS EOP SEQUENCE

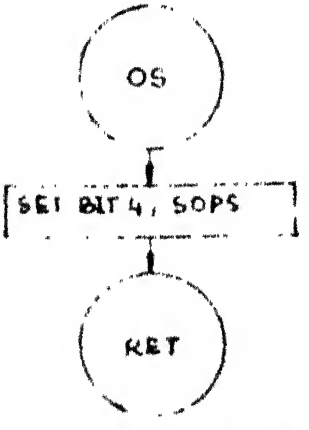


FIG: 3.66.6 ONLINE SERVICE RTN.

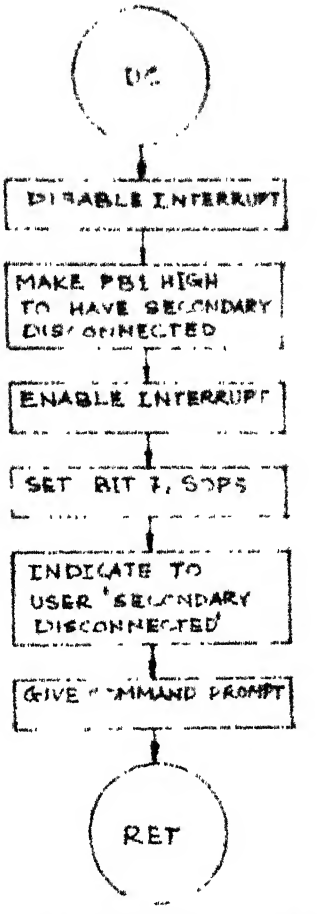


FIG: 3.66.5 DISCONNECT SERVICE RTN

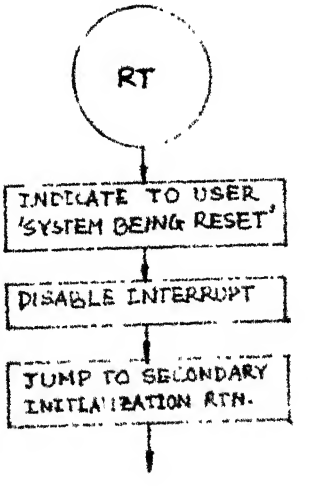


FIG: 3.66.4 RESET SERVICE ROUTINE

c) User Side Routines :

i) Command Interrupt Routine

ii) Subroutine to decide the mode of TTY response

iii) Subroutine for TTY response in Mode 0

iv) Subroutine for TTY response in Mode 1

v) Subroutine for TTY response in Mode 2

vi) Various user command routines.

1) Store, 2) Display, 3) Transmit, 4) Restore,

5) Status Poll

The user routines of the secondary are exactly identical to the corresponding routines of the primary. Hence, for their flow charts refer to Figs. 3.5c.1 to 3.5c.6.

d) Loop Secondary main program :

The secondary main program controls the execution path of the secondary software. If there is any message to be displayed and display lock is off it displays the message. Further, it checks for unprocessed message in the Rx. buffer and jumps to the message processing routine if there is any. Fig. 3.6d shows the flow chart of this routine.

Over and above these routines the controller software contains a few other utility routines (both in Primary and Secondary) which are not being included here for the sake of brevity. The routines described above constitute the intelligence of the controller.

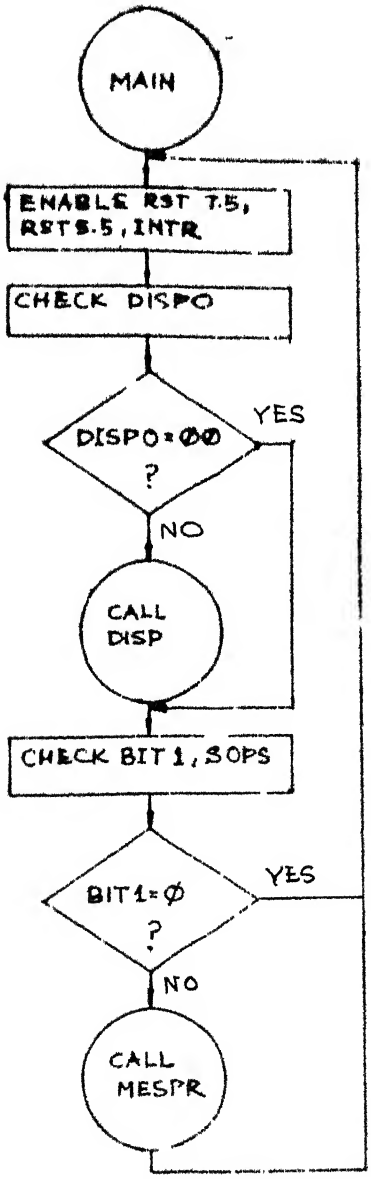


FIG: 3.6d. SECONDARY MAIN ROUTINE

CHAPTER 4

USING THE LOOP STATION FOR COMMUNICATION

4.0 This chapter describes how the network can be used by users for effective communication. Though the primary is vested with all the controlling power on all operations, all the secondary stations have equal status. So if we can think of the primary as something equivalent to a operators console station which can be operated by a privileged user only, then the network becomes perfectly fair.

4.1 POWER-ON SEQUENCE

The loop network implemented is a master-slave configuration. This implies that the Primary will have to take care of almost all the supervisory activities, involved in keeping the network operative without problems.

Initially, when the system is to be started, one should adopt the following sequence. First, all the secondaries would have to be activated. The secondary stations on power ON get initialized with their receive logic active and start listening to the loop. The Primary station is then activated and it starts the process of EOP token generation and transmission of the token round the loop. If the primary is activated first then the reception of the token character in the secondary before it is initialized, may create some erroneous condition.

4.2 LOGGIN--IN TO THE LOOP STATION

After the initialization process is over the primary responds to the user with the following message :

SDLC LOOP PRIMARY STATION

MU.P.DEV.SYSTEMS LABORATORY

ACES, IIT.K: 83-84.

- PLEASE LOGIN

It then issues a prompt to the user :

>

The user then types in a 2-character Login code, which is specific to the station. In our case the code is $N. The user follows it with a carriage return. If the Login code matches with the characters typed in, the user is allowed access to the loop station. The primary responds with a prompt :

The secondary response on power ON is exactly the same except that it notifies to the user its address in the loop which identifies it. It gives the following message.

SDLC LOOP SECONDARY STATION: XX

in the first line, where XX is its destination address. The remaining portions are exactly the same.

If the login-code does not match however, to the character typed in, the station responds with the message :

— INVALID ENTRY, TRY AGAIN

It then gives the prompt for typing in the Login code again.

>

4.3 MAKING THE SECONDARIES ONLINE

All the secondaries are initialized in Disconnect Mode. This means initially they have their transmit and receive logic externally connected. On power-on, during the initialization process the secondaries indicate their status to the user with the message :

< SECONDARY DISCONNECTED FROM LOOP >

The secondaries cannot become ONLINE unless they receive an ONLINE command from the primary.

So after the Primary user logs in and the station responds with a '.' indicating that it is now in command mode, the Primary has to give an online secondary command.

The format of the command is OS XX, where XX is the destination address. When XX = OO then the online command is sent out to all the stations in the loop (excluding the primary itself), as OO is the general address of the loop.

So to make all the station initially online, the primary station has to issue an Os OO.

On getting the OS command the secondary will make itself ONLINE and indicate to the user with a message :

<SECONDARY ONLINE LOOP>

Now the secondary transmit logic is actively connected to the loop and transmission and reception can take place.

If the Primary for certain reasons wants only to selectively enable a few of the secondaries and keep the others disabled, it has to give online commands with the specific addresses of the intended secondaries.

However, OS 01 has no meaning as Primary Station is perpetually ONLINE. If this is given, the station perform no operation. After the execution of all commands the station responds to the user with a '.' prompt to indicate that it is ready for the next command.

Any OS command in the secondary generates an error. The secondary responds with :

- COMMAND ERROR.

4.4 TRANSMISSION AND RECEPTION ON THE LOOP

Once stations selected by Primary are mode ONLINE, the transmission and reception operation can start.

To transmit a message we have to store the message first in the transmit buffer of the station. The command for storing the message is ST. The user types

. ST

On getting the ST command the station responds with a store prompt to indicate to the user that he may type in message. Whatever is typed in after this prompt is considered as data and is not exammed by the station. We show an example below :

- THIS IS A TEST MESSAGE.*

All the messages have to be terminated by an end of Text character. Here we have to give '*' to terminate a message. The station responds with :

. During the storing operation we can use the backspace key for any correction. If the message typed in becomes so long that the Tx Buffer space gets filled up, the station immediately terminates the storage mode and responds with

- END OF TX. BUFFER REACHED

. But it does not destroy the message already typed in and automatically enters the END OF TEXT character at the end of the message.

The user may like to see whether the message is properly stored. He types :

. DP

On getting DP command the station displays the message stored at the bottom of transmit queue, which is the currently typed in message. In this case it responds :

- THIS IS A TEST MESSAGE.

. Now the user may like to transmit this text to any station, 02 say. User now types :

. TX 02

On getting the TX command the sender initiates the message at the top of Tx. buffer for transmission to the test addr. 02. There are two possible reactions. If it does no transmit the frame and receive an acknowledgement or it may not and try retransmission a number of times. It does not get back an acknowledgement even after the retransmission, it responds to the user :

- RX. NONRESPONSIVE TX. ATTEMPT GIVEN UP.

. However, if it gets back an acknowledgement it simply responds with the command prompt .

. In both the cases of DP and TX, if No earlier ST command is given, which implies that there is no uninitiated message in the TX. buffer, the station responds with :

- TX. BUFFER EMPTY

. If the user wants to retransmit the earlier message to a different station it types in :

. RS

This command restores the previously stored message in the TX. buffer. Now another TX command will transmit the earlier message to the destination station.

All these commands are common to both the Primary and the secondary. One Secondary can transmit a message to any other secondary with the same TX YY command. Though the processing of the frame is entirely different and it gets routed via the primary, to the user all these operations are transparent.

If any station XX, gives a TX XX command the message simply comes back to it. This is a very useful thing so far as testing the loop is concerned. For the Primary a primary to primary transmission after power ON is useful to check that the loop is operative. For the secondary this self transmission checks out that it is indeed made ONLINE.

4.5 SOFTWARE RESET

If due to any reason a station starts malfunctioning, it can be brought back to normal operation by a software RESET command RT. While all the station can reset themselves, the Primary reserves the right of resetting any other station if it so wishes. On typing the reset command :

. RT YY

The station YY responds with :

- SYSTEM BEING RESET

Then it starts its initialization routine and after it gets initialized it responds with the usual message for loggin-in.

Note that if a secondary station is issued RT YY command, it has to be made ONLINE again, since on initialization it gets automatically disconnected from the loop.

4.6 DISCONNECTING AN ALREADY ONLINE SECONDARY

The Primary may want to disconnect a secondary which was previously made ONLINE. This becomes necessary if the secondary wants itself to be taken out of the loop or, the operation of the secondary was causing problem to the loop. The primary user types in.

. DC ZZ

where ZZ is the destination secondary.

In the secondary ZZ, the user is notified with the message :

< SECONDARY DISCONNECTED FROM THE LOOP >

4.7 STATUS POLL OF A STATION

Before transmitting to a station the sender may like to know if the receiver is disconnected/Online and whether it is being actively operated by any user. The sender types in :

. SP WW

where WW is the destination address.

The receiving station responds with a response frame if it is online, else it does not. The sending station checks the response frame if it gets any, else it times out.

In the first case the sending station responds with :

- STATION ONLINE USER OFF

OR

- STATION ONLINE USER ON

as the case may be. In the later case (no response from secondary WW) after the timeout, the sending station responds with :

- STATION DISCONNECTED

4.8 LOGGING OUT FROM THE STATION

After the transmission and receptions are over the user may like to terminate the command mode and leave. He types in:

. $Q

On getting this command, the station terminates the command mode and waits for the next user. It responds with :

- LOGIN FOR REENTRY

>

Thus we see that the user without having to actually manipulate the transmission, reception operations can effectively communicate with another user. It needs to know only the destination address for any transmission operation. The rest of the lower level activities is managed by the station transparently to the user.

CHAPTER 5

CONCLUSION

5.0 THE ENVIRONMENT IN WHICH THE SYSTEM WAS IMPLEMENTED

We briefly describe here the actual environment in which this system was developed. As we have described elsewhere, we concentrated mainly on the design of the Access controller to the loop net (or the Interface Message Processor as it is better known). For setting up the network, we used as user interface the dumb terminal developed in our department. This dumb terminal sends out an ASCII code corresponding to any key pressed on the keyboard. It also displays the character whose ASCII code is received by it on the CRT screen. Because of the limited availability of these terminals we were unable to set up more than two stations (which is the bare minimum) for the network. The network set up contained one Primary and one secondary device. However, for the purpose of testing it would have been desirable to set up at least three (with 2 secondaries) to guage the general performance of the network. In any case, we could check out all the options designed successfully on our set up itself. Since the software for the controller was written in a general way, we are confident that the network would be reliable even with more number of secondaries.

Another factor that could not be checked out was the geographical performance of the network. Our set up was located within a radius of a few metres. However, setting up a network over a longer distance is quite straightforward. Since this is a local network we do not need any modulator/demodulator stages and the transmission is at the baseband. So for transmission over longer distance we only need some booster stages at the output to provide sufficient current drive for a long line of cable. The ICs 75107, 75109 transmitter receiver pairs can be used as the current drivers upto a distance of 500 metres. For longer distances we need to add repeater stations; this, however, was not attempted in this project.

5.1 SUGGESTION FOR MODIFICATIONS AND IMPROVEMENTS

a) As mentioned in the chapter on hardware design, the Primary and the Secondary hardware have been designed to be as nearly identical as possible. Here we suggest the slight modification in hardware necessary to make the same hardware programmable both as Primary and Secondary.

The differences between the Primary and Secondary hardwares are only at two points :

1) In the connection of the loop clock - In case of Primary, there is a separate source clock for the loop. But in case of secondaries, no such source clock is needed. The secondaries

generate the clock from received data, by means of their internal DPLL so in their case Both pins $\overline{TXC}$, and $\overline{RXC}$ are connected to $\overline{DPLL}$.

2) In the RST 6.5 connection - The secondaries do not need the EOP timeout interrupt (RST 6.5) as it does not generate the EOP token. So in case of secondaries the RST 6.5 would be low and not connected to the timer output which generates the interrupt, as in Primary.

By using one of the Port B outputs of the 8273 in conjunction with some additional NAND gates we can very easily make the hardware programmable, so that it satisfies the appropriate conditions in each of the cases. Fig. 5.10 illustrates how this can be done using the PB2 output of the 8273. Referring to the figure we see that RST 6.5 = OUT2.PB2. If PB2 = 0, RST6.5 = 0, that is Lo. If PB2 = 1, RST6,5 = OUT2 as is required. Secondly, $\overline{TXC}$ = PB2.OUT1 + PB2.($\overline{RXC}/\overline{DPLL}$). If PB2 = 0, $\overline{TXC}$ = $\overline{RXC}/\overline{DPLL}$ and if PB2 = 1, $\overline{TXC}$ = OUT1. Thus programming PB2 = 0 for secondary operation and PB2 = 1 for primary operation would serve our purpose. This must be done in the initialization routine of the respective stations.

b) A logical extension of this network is to implement a gateway for connecting to a second similar network. We have in fact designed the possible control codes that will supervise communication between two loops through the gateway (Ref. to Table 1.40). However, this was not implemented.

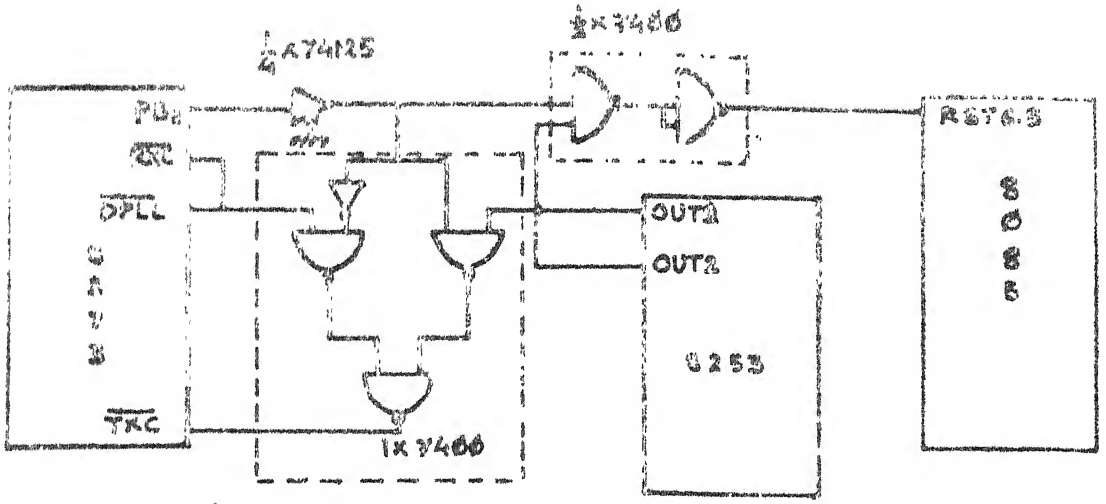


FIG: 5.10 THE SCHEMATIC DIAGRAM SHOWING HOW THE PRIMARY AND SECONDARY OPERATION CAN BE PROGRAMMED BY PB2 OUTPUT.

A possible scheme for the gateway is to have two 8273's in back to back configuration. They would share the same memory buffers. The transmit buffer of one should be the receive buffer of the other and vice versa. Each one would listen to one of the two loops. Whenever a frame having a control code 8C is received, it would accept the frame, change the code to OC and store it in its receive buffer, which is the transmit buffer of the other 8273. On the transmit side, it would look for a message in its transmit buffer for any message received and stored by the other 8273. If it finds a message, it would transmit that frame, following receipt of an EOP character in its receive logic. The 8273 listening to the other loop would perform exactly the complementary operations. Thus if the sending Primary station in loop A transmits a frame with code 8C, it would be received by the receiving Primary station in loop B as a frame with code OC. Similarly, if the sending Primary station in loop B transmits a frame with code 8C, it would be received as a frame with code OC in the Primary of loop A. Acknowledgement frames would be routed in exactly the same way through the gateway, their code being switched from 8D to OD. The gateway can be issued general receive commands so that no loop address on either side is wasted. The gateway would accept or reject a frame by checking its control code only.

c) Another possible extension is to have multiple users connected to the same IMP. We must use an individual serial link (via 8251) for each of the users. We can then serve the different users through different interrupts or we can make provision for reading the identity of the interrupting users through some port and serve them through the same interrupt. The 8273 does not support double byte destination address. We can take care of that by using one byte of data as the implied address. In this way we would be able to access all the users connected to the same IMP individually as well as collectively. The number of associated control codes can be generated to supervise such operations. In case such an extension is undertaken, the amount of RAM would require a drastic increase. The future memory expansion space provided in the system can be utilized for this purpose.

d) The speed of this system can be improved from 9.6 Kbaud to 64 Kbaud, by using an external clock recovery circuit rather than the internal DPLL of the 8273 itself. The using of Intel 8274, multiprotocol controller chip, instead of the present 8273 would enhance the speed of the network drastically to 880 Kbaud, without any considerable change in the hardware.

In conclusion, we can say that we have been able to implement the basic structure of the controlling station so that transmission and reception can be carried out smoothly. Some

supervisory operations are also provided for the proper and effective monitoring of the loop. Further commands can be added to improve the supervisory activities on one hand and to make the station more user friendly on the other. We advise that such a scheme is undertaken so that the network can actually be put into operation for the users.